NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

CORRELATION AND KALMAN FILTER TRACKING SOLUTIONS FOR THE NRL NIPEX ALGORITHM

by

Lee Kurt Allred

September, 1995

Thesis Advisor: Robert G. Hutchins

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1995	Master's Thesis

4. TITLE AND SUBTITLE: CORRELATION AND KALMAN FILTER TRACKING SOLUTIONS FOR THE NRL NIPEX ALGORITHM

5. FUNDING NUMBERS

6. AUTHOR(S): Lee Kurt Allred

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis presents an algorithm designed to correlate and track airborne targets using positional and identification data reported from the Naval Research Laboratory Integrated Proforma Exploitation (NIPEX) test bed. The algorithm processes bearing and range inputs using an Interacting Multiple Model Kalman Filter (IMMKF). Association and gating of input measurements is accomplished with; (1) logical functions using amplifying information from NIPEX output data, (2) linear velocity checks on all associations, and (3) a scoring method based on the Mahalanobis distance comparing positional inputs to Kalman Filter predicted states. Tracking is accomplished in two-dimensional space and performance of the algorithm is demonstrated for simulated data as well as actual collected data provided by the Naval Research Laboratory (NRL). The compatibility of the algorithm for fusion with additional sources is also addressed.

14. SUBJECT TERMS Kalman Filter; IMMKF, target tracking; NRL NIPEX Algorithm

15. NUMBER OF PAGES 74

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

CORRELATION AND KALMAN FILTER TRACKING SOLUTIONS FOR THE NRL NIPEX ALGORITHM

Lee Kurt Allred
Lieutenant, United States Navy
B.S., University of Utah, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1995

Author: _____
Lee Kurt Allred

Approved by: _____
Robert G. Hutchins, Thesis Advisor

_____
D. Curtis Schleher, Second Reader

_____
Fred H. Levien, Chairman
Electronic Warfare Academic Group

ABSTRACT

This thesis presents an algorithm designed to correlate and track airborne targets using positional and identification data reported from the Naval Research Laboratory Integrated Proforma Exploitation (NIPEX) test bed. The algorithm processes bearing and range inputs using an Interacting Multiple Model Kalman Filter (IMMKF). Association and gating of input measurements is accomplished with; (1) logical functions using amplifying information from NIPEX output data, (2) linear velocity checks on all associations, and (3) a scoring method based on the Mahalanobis distance comparing positional inputs to Kalman Filter predicted states. Tracking is accomplished in two-dimensional space and performance of the algorithm is demonstrated for simulated data as well as actual collected data provided by the Naval Research Laboratory (NRL). The compatibility of the algorithm for fusion with additional sources is also addressed.

TABLE OF CONTENTS

I. INTRODUCTION

A. PURPOSE

The purpose of this thesis is to develop a Kalman Filter based correlation and tracking algorithm designed to establish, maintain and smooth target tracks using output data from the Naval Research Laboratory Integrated Proforma Exploitation (NIPEX) test bed. The correlator-tracker must be able to handle both maneuvering and non-maneuvering targets existing in airspace densities consistent with both military and civilian activity. Due to the dynamic nature of these environments, the expectations for the accuracies of the tracking algorithm must be commensurate with input data and its update rates. In short, this thesis will review the output data available from the NIPEX algorithm, establish the performance requirements necessary for the Kalman Filters, and outline a viable solution for providing a robust target tracking system.

B. ORGANIZATION OF THESIS

This thesis is organized into six parts. Chapter II begins by providing an overview of the Naval Research Laboratory (NRL) developed NIPEX system and the basic concepts involved in passively exploiting air traffic control (ATC) and information friend or foe (IFF) interrogations and transponder replies. Exploitation of these radar signals by the NIPEX system provide the aircraft position, altitude and identification information used as inputs for the algorithms presented in this thesis. Chapter II also discusses the format and associated error variances of the output data from the NIPEX system. Chapter III describes the logic flow of input data to the correlator-tracking algorithm and the various gating criteria used to route positional measurements to the Interacting Multiple Model Kalman Filters (IMMKF). The IMMKF equations will be discussed in Chapter

IV, along with their associated design parameters. Chapter V follows with an evaluation of the correlator-tracking algorithm, including performance versus maneuvering and non-maneuvering targets. Conclusions are outlined in Chapter VI and include; a summarization of the author's opinions as to success of the correlator-tracker to meet the needs of the NIPEX system, recommendations for possible improvements, and the algorithms compatibility for possible fusion with other sources.

II. BACKGROUND

A. OVERVIEW OF THE NIPEX SYSTEM

The NRL Integrated Proforma Exploitation (NIPEX) test bed is a system that passively determines aircraft position, altitude and identity by intercepting signals from an unwitting (victim) ground based radar, it's associated identification friend or foe (IFF) or air traffic control (ATC) interrogator, and the airborne transponders that reply. [Ref. 1] The signals that are exploited by the NIPEX system are air traffic control (ATC) signals used by the aviation community world wide as defined by the International Committee on Aviation Organization (ICAO). All aircraft within commercial air space are required to be equipped with transponders that reply to specific pulse burst patterns transmitted by ATC interrogators.

Exploiting ATC interrogations and transponder replies as opposed to reflected radar signals offers a few advantages. Intercepting signals transmitted directly by the aircraft instead of reflected radar energy, provides an inherently greater received signal strength, and therefore a more simplified receiver system is required. Also, as far as stealth issues are concerned, signal reception is not affected by low radar cross section aircraft, and the NIPEX system is not vulnerable to detection or exploitation from any of its own radiations. A disadvantage, however, is that a transponder can be turned off by the aircrew if they do not wish it to radiate (even though such action may be in violation of commercial airspace requirements).

ATC interrogator pulse groups may request any one of four reply modes: 1, 2, 3/A ("squawk" number), or C (altitude). Modes 1 and 2 are used exclusively by military entities but mode 3/A and C are used both by western military systems and the entirety of commercial aircraft. The mode an interrogator

requests is determined by spacing between the leading and trailing pulses of the interrogation pulse group (i.e. the framing pulses). Reply data bursts do not explicitly indicate the type of information contained within them (altitude, mode 3/A, or other) but there are implicit indications which are used in the NIPEX algorithm. [Ref. 1] First of all, the NIPEX algorithm assumes only altitude or mode 3/A queries are being transmitted by the interrogator. These are the primary modes interrogated by land based ATC/IFF systems designed for use with civilian as well as military aircraft. Secondly, valid altitudes are represented only in increments of 100 feet. Therefore, the NIPEX algorithm declares a reply to be an altitude only if it decodes to a valid altitude value. Otherwise, it is assumed to be a mode 3/A "squawk" number. Altitude values are translated from pulse position coding via a look-up/matching table. Mode 3/A squawk numbers are translated directly from a pulse position coding scheme and may be any combination of four sequential octal numbers (i.e. none of the digits can be above seven: 7777). Obviously, this approach will not work if the additional military modes (modes 1 and 2) are interrogated or if a mode 3/A squawk number corresponding to a valid altitude has been assigned. More robust techniques have also been explored for monitoring the interrogations being sent to the aircraft (which explicitly request the information desired). This method attempts to marry the transponder responses to specific interrogation requests. Redundancy and consistency checks are also used to increase the accuracy of which decoding scheme is to be used.

An ATC/IFF interrogator is normally co-located with a ground based rotating ATC radar and has a directional beam pattern. The interrogator transmits electronic queries ("interrogations") through its rotating directional antenna to elicit responses from aircraft transponders in the illuminated region. Interrogator side lobe suppression (ISLS) techniques are used to prevent aircraft

not in the main beam of the interrogator from replying. Valid ATC interrogations consist of the two framing pulses and a third ISLS pulse contained within the bracketing pulses. The framed ISLS pulse is transmitted at a signal level smaller than the main lobe but larger than the side lobes of the interrogator to inhibit replies from directions not in the main beam (only signals received within the main beam will have framing pulses of signal strength greater than the ISLS pulse). [Ref. 1]

Aircraft transponders have both a receiver and a transmitter to reply via omnidirectional antennas after receiving valid interrogation pulse bursts. When a transponder receives an interrogation it waits for a specified delay period (21 microseconds) and then replies with a pulse burst containing the requested data in a pulse position coded format. The transponder replies also have two framing pulses and a binary coding scheme of 13 possible pulses in between the framing pulses to relay the requested data.

Because all transponders are attempting to reply to all interrogations, ambiguity can arise as to which replies correspond to which interrogations. However, an interrogator can determine which replies it has elicited by finding those replies which share synchronous time differences from its own interrogations. This process of finding synchronous replies is called "defruiting". [Ref. 1] Once the replies are sorted out, individual aircraft positions (bearing and range) from the victim interrogation radar can be determined.

The NIPEX system operates from a remote location within line of sight of the victim interrogation radar system so it can slave itself to the scan rate of a victim interrogator and to the times of the interrogation transmissions. This enables it to determine precisely when (with 100 nanosecond resolution), and in which direction a particular interrogation was transmitted. After the system filters out asynchronous transponder replies, it uses bistatic radar timing

relationships (between interrogations and replies) along with a priori knowledge of the transponder processing delays and victim radar system location to compute range and azimuth to aircraft. [Ref. 1] Therefore, the system determines aircraft positions without any direction finding antennas.

B. BISTATIC RADAR GEOMETRY

Bistatic radar equations are a key element of the NIPEX system. The NIPEX system assumes the role of the receive antenna in the bistatic radar equations. [Ref. 2] Figure 2.1 demonstrates a typical two-dimensional

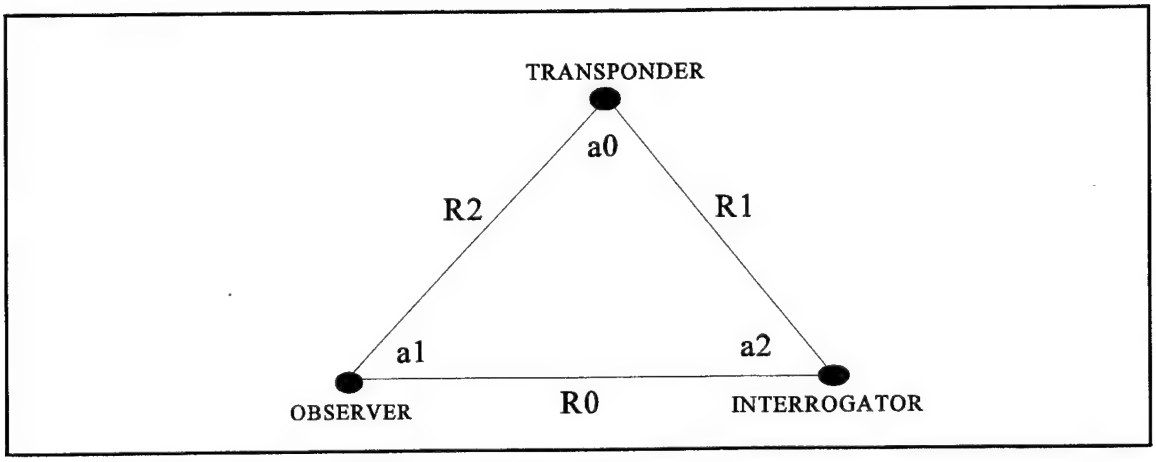

Figure 2.1 The Bistatic Triangle [Ref. 1]

geographic representation of the "bistatic plane", at an instant in time, along with locations of the radar interrogator, aircraft transponder and observer (NIPEX system). The geometry assumes a flat earth model that ignores the curvature of the earth. The distances ***R0***, ***R1***, and ***R2*** are the line of sight distances from the interrogator to the transponder, from the transponder to the observer, and from the observer to the interrogator, respectively. The angles ***a0***, ***a1***, and ***a2*** are subtended angles from the vertices at the transponder, the observer, and the interrogator, respectively. [Ref. 1]

Using geometric and trigonometric relationships, it can be shown that range (***R2***) and azimuth (***a1***) from the observer (the NIPEX system) can be written in terms of the parameters, ***R1+R2***, ***R0***, and ***a2***, which are measurable quantities (***R1*** is also used but is simply the combined distance, ***R1+R2***, minus the range, ***R2***). These calculations are valid for any relative position of the three elements in the bistatic plane (observer, interrogator, transponder) except when a transponder lies directly between the observer and the interrogator. For these instances, the geometric and trigonometric methods used for target location yield ambiguous results and consequently the NIPEX algorithm ignores replies from these targets. [Ref. 1]

$$Range(R2) = \frac{(R1 + R2)^2 - 2R0(R1 + R2)\cos(a2)}{2((R1 + R2) - R0\cos(a2))}$$

$$Azimuth(a1) = \arcsin\left[\frac{R1\sin(a2)}{R2}\right]$$

The combined distance, ***R1+R2***, is derived from the a priori knowledge of the distance ***R0*** added to the transponder processing delay and the time of arrival (TOA) measurements from the interrogator and transponder multiplied by the speed of light, "c".

$$(R1+R2) = c(TOA_{transponder} - TOA_{interrogator} - transponder\ processing\ delay) + R0$$

Target aircraft azimuth relative to the interrogator is determined based on the pointing angle of the radar antenna when transponder replies are received. The time period between receiving the last interrogation signal and a transponder

reply can be translated to the angle ***a2*** by multiplying it with the scan rate. The transponder processing delay is theoretically included, but becomes negligible for this calculation due to the relatively slow antenna rotation rates. The scan rate for the rotating radar is determined from observing the time delay between successive interrogations pointed at the observer site.

$$a2 = (TOA_{interrogator} - TOA_{transponder})\left[\frac{360^{o}}{Scanperiod}\right]$$

C. DETERMINATION OF MEASUREMENT VARIANCES

Determining the accuracy and variances associated with the bearing and range "measurements" provided by the NIPEX system is necessary to establish the covariance matrices in the Kalman Filter equations. The covariance matrices represent the statistical measure of uncertainty in the provided measurements and affect the accuracies of the Kalman Filter track updates. Therefore the tracking capabilities of the Kalman Filter are influenced by the measurement variances input to it. All NIPEX calculations for aircraft bearings and ranges are based on timing relationships, and all time of arrival (TOA) measurements are limited by a 100 nanosecond (nsec) measurement resolution of the signal sampler. The three primary measured components that make up the bearing and range calculations are ***R0***, ***R1+R2***, and ***a2***. [Ref. 1]

The base distance, ***R0***, between the observer (NIPEX system) and the interrogator is assumed to be constant and determined with the accuracy of Global Positioning Systems (GPS). The accuracy of this system will be assumed to be accurate to within 15 meters. Therefore maximum errors should be 30 meters or less for the base distance, ***R0***. The combined distance, ***R1 + R2***, is based on two time of arrival (TOA) measurements and the base distance, ***R0***.

The two TOA measurements are accurate to within 100 nsec each, multiplying by the speed of light yields 30 meters of possible distance error for each TOA measurement. For the purpose of this research, the error ranges are assumed to be "2σ" values, which would include 95% of the errors in a Gaussian distributed error population. This assumption yields an error variance for ***R0*** of 225 meters squared, and an error variance of 675 meters squared for the combined distance, ***R1+R2***, where the total variance is the sum of its components ($\sigma^2_{total} = \Sigma\sigma^2_{components}$).

The angle, ***a2***, is based on three TOA measurements. The first two TOA's determine the scan rate and the third TOA determines the time period between the interrogator antenna pointing at the observer and the antenna pointing at the transponder. This defines the angle between the observer, the interrogator and the transponder aircraft. Unfortunately, the accuracy to which the scan period can be measured is also a function of pulse group repetition interval (PGRI), which can vary substantially from one interrogator to the next. The largest PGRI tested against the NIPEX system was 4786 microseconds (μsec). [Ref. 3] Assuming that twice this value is the "2σ" error range for time measurements, its error variance would be 22.9 μsec. This variation is significantly larger than the TOA resolution and, therefore, becomes the dominant factor. Interrogator antenna rotating scan rates vary, but primarily occur between 36 and 90 degrees per second. Multiplying the faster rate by the maximum time variance yields an angular variance in ***a2*** of 0.0021 degrees squared.

To determine range and bearing measurement variances, calculations were attempted using Taylor series expansions of the equations for ***R2*** and ***a2***. Because these calculations are geometry dependent, estimates for range and bearing variances were made from the Taylor series expansions and estimated variations in ***R0***, ***R1+R2***, and ***a2***. Measurement variances of 1600 meters

squared for ranges and one degree squared for bearings were used in the correlator-tracker algorithm. Azimuth variation was the most difficult parameter to estimate and has the potential to introduce the greatest measurement variances at large ranges. Therefore, an evaluation of the data produced by the correlator-tracker was conducted to validate the azimuth variance estimate used. Calculations based on the collected data show the azimuth variance between NIPEX reported positions and their associated IMMKF generated tracks to average less than 0.2 degrees. If only a portion of this variance is attributed to measurement noise, this evaluation provides credibility to the variance estimates used in this algorithm.

D. MEASUREMENT DATA FORMAT

The output data from the NIPEX algorithm used by this correlator-tracker is formatted as an ASCII text file ("datalog.txt"). Each "measurement set" from the NIPEX system computations includes, at a minimum: time of arrival (TOA) of the reply pulse burst, bearing in degrees and range in kilometers of an aircraft from the NIPEX system or from the victim radar system. Also, the altitude or mode 3/A identification number may be present, if available and decoded properly in the NIPEX algorithm. A sample of the data collected is included in Appendix A. The "read" function of this algorithm is set to read in only four columns of data; time, mode 3/A, bearing in degrees, and range in kilometers. Additional information included in the output text file is not used by this correlator-tracker. Because of the logic associated with the "read" function, alterations to the format of the output text file would have to be accounted for in the correlator-tracking algorithm.

III. DATA ASSOCIATION AND TRACKING METHODOLOGY

A. BACKGROUND

The NIPEX system provides a "measurement" of bearing and range, along with target identification (in the form of a mode 3/A squawk number), for each target during each sweep cycle of the victim interrogator radar. The geometric data (bearing and range) are derived quantities, computed by the NIPEX algorithms, but are treated as the basic measurement data in the association and tracking algorithm developed for this research. Altitude information may also be reported in aircraft transponder replies but this data is not used for association or tracking purposes in the algorithm developed for this research. The target identification information can be subject to improper decoding errors, but is considered quite reliable based on the collected data supplied by NRL. Therefore, the data association strategies employed here assume a low percentage of erroneous mode 3/A squawk numbers.

Target track state estimates consist of two-dimensional position and velocity estimates. Other information associated with each system track includes state covariance matrices, modal likelihood probabilities, and the time of last update (the time at which a measurement was last associated with the track). The modal probabilities are required by the interacting multiple model (IMM) algorithm, which is employed to track maneuvering targets. The basic logic used to predict future target states and to update state estimates with new measurement data is the Kalman Filter. Data association algorithms, Kalman Filters and IMM methods are discussed in detail below.

Because the IMMKF design is based in a Cartesian coordinate system, bearing and range measurements must be converted to Cartesian coordinates. For the purpose of velocity association gating, this conversion process is done with

conventional methods. However, after associations are made to active tracks, measurement inputs to the IMMKF are converted via a debiasing compensation method. This process will be explained further in Chapter IV.

B. ALGORITHM DEVELOPMENT

A review of the collected data set from the NIPEX system and the goals of the correlator-tracker resulted in an algorithm designed to provide a level of performance relative to the input data accuracy, and as uncomplicated routing logic as possible for data association. Because of the update rates of the NIPEX reported data, track quality can not be seriously attempted beyond a "flight following" level of performance. High speed turns and simultaneously crossing aircraft tracks are difficult problems for nearly every tracking system and could induce tracking errors in this Kalman Filter based algorithm as well. A dynamic environment like that which most highly maneuverable military aircraft operate within will be the greatest test for this algorithm.

The first step in the problem of tracking multiple targets is the gating and association of measurement inputs to active or potential tracks. This is accomplished in many tracking systems using just the positional data in comparison to active or potential track state predictions. However, the data available from the NIPEX system can include, in addition to positional data, amplifying information regarding an aircraft's reported altitude or mode 3/A squawk number (mode 3/A data was available approximately 70% of the time and altitude data was available approximately 90% of the time in the data set collected by NRL). The primary goal in using either of these pieces of additional information would be in providing quick and more accurate data association. With this in mind, the mode 3/A squawk number was chosen as the most expeditious method of associating measurement data to tracks. Neither the

altitude nor mode 3/A squawk number has a higher inherent probability of being accurate more often than the other. And even though the altitude information appears more often than the mode 3/A number in the collected data, altitude gating would require significantly more complicated association techniques and additional processing time in the algorithm. Therefore, mode 3/A was chosen as the simplest, and potentially most accurate method of sorting and associating new data to existing or potential tracks.

The general method of association used by this algorithm is one of "measurement-to-track" correlation. When a new measurement is received, the algorithm attempts to correlate it to an existing active or potential track (pretrack) to be used to update the estimate of the track's current state. Active tracks take priority for matching over pretracks. If no association can be made for either active tracks or pretracks, the new measurement is converted to pretrack status and associated with a track identification number (either its mode 3/A number or an arbitrarily assigned four digit number).

Mode 3/A squawk numbers are used in the correlator-tracking algorithm as a primary logic variable for sorting and associating measurements. Measurement data with an associated mode 3/A number is routed for matching with active or potential tracks which also carry the same mode 3/A number. Linear velocity checks are computed for all mode 3/A "matches" to reduce the possibility of an erroneous measurement to track assignment. The mode 3/A squawk number is used throughout the algorithm for the track identification numbering as well. Even measurement data that does not have an associated mode 3/A number associated with it is assigned a four digit identification number (although not a valid mode 3/A number). This number is used strictly for the purpose of track and pretrack identity, associations are not made based upon these arbitrarily assigned numbers.

When association of measurements to active tracks or pretracks can not be accomplished via mode 3/A numbers, a hierarchical system of gating techniques is used. The simplest gate employs linear velocity checks. These are computed to associate measurement data to pretracks without a valid mode 3/A number because pretracks do not have state estimates associated with them yet. The second, and more complex method of association gating involves the use of Mahalanobis distance calculations. It compares new measurements to active tracks using the active track state estimates and computes a "goodness of fit" statistic for each possible track association. It then chooses the best fit for association and track state update calculations. These techniques will be explained in more detail in the following section.

C. ASSOCIATION AND GATING TECHNIQUES

1. Velocity Gating

Linear velocity checks are computed throughout the algorithm to ensure updated track speeds will be "reasonable" before a measurement is assigned to a track. Essentially, the position of the measurement is compared to the last position of the active track or pretrack and a straight line distance is computed between them. Then the time of the last state estimate is subtracted from the measurement time and this time difference is divided into the straight line distance to yield a speed in meters per second. This speed is accepted if it is between 30 and 350 meters per second (which equates to approximately 60 to 680 nautical miles per hour or "knots") and can be adjusted if so desired. Speeds that are rejected initiate algorithm logic to search for a more suitable track assignment.

2. Mahalanobis Distance Association

Mahalanobis distance association is a minimum distance classification method. [Ref. 4] When a new measurement fails to correlate with any active tracks via mode 3/A, an attempt is made to correlate the measurement with an active track via Mahalanobis distance association. A Mahalanobis distance (D^i) is computed for each of the three modal likelihoods of each active track based on a "goodness-of-fit" of the measurement to the predicted state for each modal likelihood. The minimum Mahalanobis distance for each track is then compared to the other minimum Mahalanobis distances from each of the other active tracks to determine the best fit (the minimum value).

$$D^i = [\underline{z} - H\underline{\hat{x}}^i]^T [H\hat{p}^i H^T + R]^{-1} [\underline{z} - H\underline{\hat{x}}^i]$$

The Mahalanobis distance is a statistic assumed to have a Chi squared distribution and, therefore, a critical value is determined for four degrees of freedom at a significance level of 99%. If the minimum Mahalanobis distance is less than the critical value (14.9), an association is made. Otherwise the algorithm logic downgrades its search to look for a velocity match with a pretrack.

IV. KALMAN FILTER DEVELOPMENT

A. THE DISCRETE KALMAN FILTER

The purpose of a Kalman Filter is to estimate a state vector at the time of the last measurement based on all past measurements. Prediction refers to the estimation of a state vector at some future time, and smoothing refers to the estimation of a state vector at some previous time based on all previous measurements taken up to the present time. All of these functions are closely related in that an estimate, $\underline{\hat{x}}$, is a computed value of a quantity, $\underline{x}$, based upon a set of measurements, $\underline{z}$. Unbiased estimates have an expected value equal to that of the quantity being estimated. Minimum variance unbiased estimates have the smallest error variance of any other unbiased estimate. Consistent estimates converge to the true value of $\underline{x}$ as the number of estimates increases. Therefore, optimal estimators would be unbiased, minimum variance, consistent estimators. [Ref. 5]

The discrete time Kalman Filter system equations can be modeled as follows for measurements received at time t_k,

$$\underline{x}_{k+1} = \Phi_k \underline{x}_k + \underline{w}_k \qquad \underline{z}_k = H\underline{x}_k + \underline{v}_k$$

where Φ_k is the state transition matrix, $\underline{w}$ is the Plant noise associated with a target and is assumed to be zero mean, white and Gaussian with covariance "Q_k" ($w_k \sim N[\underline{0}, Q_k]$), H_k is a constant matrix related to the number of dimensions observed, and $\underline{v}_k$ is additive measurement noise also assumed to be zero mean white Gaussian with covariance "R_k" ($v_k \sim N[\underline{0}, R_k]$). For a target moving in a straight line at a constant speed in a two-dimensional plane, the state vector at time, t_k, is

$$\underline{x}_k = [\textit{x-position},\ \textit{x-velocity},\ \textit{y-position},\ \textit{y-velocity}]^T$$

and the associated state transition matrix is

$$\Phi_k = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where ***dt*** is the time step increment. For position estimates that are measured directly,

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The prediction step then attempts to predict the state estimate at a future time, t_{k+1}, based on all measurements up to the present time. This prediction will also have a Gaussian distribution with a covariance $P_{k+1/k}$ ($\underline{\hat{x}}_{k+1/k} \sim N[x_{k+1}, P_{k+1/k}]$) modeled by the equations,

$$\underline{\hat{x}}_{k+1/k} = \Phi_k \underline{\hat{x}}_{k/k} \qquad P_{k+1/k} = \Phi_k P_{k/k} \Phi_k^T + Q_k$$

The Kalman Filter measurement update equations for the next measurement time, t_{k+1}, are represented by the equations,

$$\underline{\hat{x}}_{k+1/k+1} = \underline{\hat{x}}_{k+1/k} + K_{k+1}[\underline{z}_k - H_k \underline{\hat{x}}_{k+1/k}]$$
$$P_{k+1/k+1} = [I - K_{k+1} H_{k+1}] P_{k+1/k}$$

where "K" is the Kalman Gain Matrix and is calculated from,

$$K_{k+1} = P_{k+1/k}H_{k+1}^{T}[H_{k+1}P_{k+1/k}H_{k+1}^{T} + R_{k+1}]^{-1}$$

and "I" is an identity matrix of the appropriate dimensions.

The purpose of this section is to familiarize the reader with the basic discrete time Kalman Filter equations before presenting the more advanced Interacting Multiple Model Kalman Filter (IMMKF) equations in the following sections.

B. THE INTERACTING MULTIPLE MODEL KALMAN FILTER

1. System Equations

The IMMKF, as outlined in *Multitarget-Multisensor Tracking: Principles and Techniques* [Ref. 6], is a hybrid filter system comprised of a finite number of system models. This particular tracking algorithm uses three system models: a straight line motion model, a left turn model, and a right turn model. State and covariance estimates are calculated and maintained for each mode and then mixed via a Markov state transition probability matrix. The end result is an overall state and covariance matrix which provides a mode conditioned combination of the latest state estimates and covariances.

Each of the three models in the IMMKF require their own set of Kalman Filter system equations. The individual system equations differ only in their state transition matrices, so all three can be modeled as follows,

$$\underline{x}_{k+1} = \Phi_k\underline{x}_k + \underline{w}_k \qquad \underline{z}_k = H_k\underline{x}_k + \underline{v}_k$$

where $\underline{w}$ and $\underline{v}$ exist as previously defined, and the following Φ_k's are used where appropriate for straight line motion or for turns. [Ref. 7]

$$\Phi_k^{straightline} = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \Phi_k^{turn} = \begin{bmatrix} 1 & \frac{\sin(\omega dt)}{\omega} & 0 & \frac{\cos(\omega dt)-1}{\omega} \\ 0 & \cos(\omega dt) & 0 & -\sin(\omega dt) \\ 0 & \frac{2\sin(\omega dt)^2}{\omega} & 1 & \frac{\sin(\omega dt)}{\omega} \\ 0 & \sin(\omega dt) & 0 & \cos(\omega dt) \end{bmatrix}$$

An angular turn rate (ω) of 0.25 radians per second is used in these matrices (which equates to "3-g" turns at 250 knots and "6-g" turns at 450 knots) but can be adjusted to the maximum anticipated turn rate for a particular target environment. Straight line motion tracking can be degraded when larger turn rates are used, however. A positive ω is used for the left turn model and a negative ω is used for the right turn model. The time difference (dt), existing throughout the state transition matrices, is calculated for each individual track update by subtracting the time of the last update from the time of the new measurement update.

2. The Filtering Process

Each of the three models must complete the following series of computations; an interaction step, a prediction step, and a measurement update step. A measurement conversion process is also done prior to the measurement update step but this computation is done only once per update cycle and then used for each of the modal update steps. After these computations are completed for each model, the mode probabilities and mixing probabilities are updated to compute the combined state and covariance estimates.

The interaction step modifies the transition possibilities from each state by multiplying the Markov chain probability matrix elements by the last modal state

probability vector elements (normalized). [Ref. 6] Using these modified modal likelihood values, an intermediate state vector is calculated for the prediction step. For example, the intermediate state vector and covariance matrix for all straight line motion possibilities ($\underline{\hat{x}}^{straight}$) are calculated from the following equations;

$$\bar{c} = P_{11}\mu^1_{k-1} + P_{21}\mu^2_{k-1} + P_{31}\mu^3_{k-1}$$

$$\underline{\hat{x}}^{straight}_{k-1/k-1} = \underline{\hat{x}}^1_{k-1/k-1}\left[\frac{P_{11}\mu^1}{\bar{c}}\right] + \underline{\hat{x}}^2_{k-1/k-1}\left[\frac{P_{21}\mu^2}{\bar{c}}\right] + \underline{\hat{x}}^3_{k-1/k-1}\left[\frac{P_{31}\mu^3}{\bar{c}}\right]$$

$$\begin{aligned}\hat{p}^{straight}_{k-1/k-1} &= \mu^1_{k-1}\left[\hat{p}^1_{k-1/k-1} + [\underline{\hat{x}}^1_{k-1/k-1} - \underline{\hat{x}}^{straight}_{k-1/k-1}][\underline{\hat{x}}^1_{k-1/k-1} - \underline{\hat{x}}^{straight}_{k-1/k-1}]^T\right] \\ &+ \mu^2_{k-1}\left[\hat{p}^2_{k-1/k-1} + [\underline{\hat{x}}^2_{k-1/k-1} - \underline{\hat{x}}^{Lturn}_{k-1/k-1}][\underline{\hat{x}}^2_{k-1/k-1} - \underline{\hat{x}}^{Lturn}_{k-1/k-1}]^T\right] \\ &+ \mu^3_{k-1}\left[\hat{p}^3_{k-1/k-1} + [\underline{\hat{x}}^3_{k-1/k-1} - \underline{\hat{x}}^{Rturn}_{k-1/k-1}][\underline{\hat{x}}_3(k-1/k-1) - \underline{\hat{x}}^{Rturn}(k-1/k-1)]^T\right]\end{aligned}$$

The prediction step uses the modified modal likelihood values and the intermediate state vectors to compute predicted state vectors and covariance matrices for all three modes. These predicted states are used later in the measurement update step.

$$\underline{\hat{x}}^i_{k/k-1} = \Phi_i \underline{\hat{x}}^{intermediate}_{k-1/k-1} \quad \hat{p} = \Phi_i \hat{p}^{intermediate}_{k-1/k-1} + Q_k$$

The measurement conversion process from polar coordinates (bearing and range) to Cartesian coordinates (which the IMMKF uses) involves a debiasing compensation to reduce additional errors in the Kalman filtering process. This "debiasing compensation" [Ref. 6] computes an average true converted measurement bias, and an average true converted measurement covariance based on the measured positions. The average true converted measurement bias is subtracted from the standard polar-to-Cartesian coordinate conversion to ensure

coordinate conversion consistency. The average true converted measurement covariance more accurately accounts for measurement variances and makes the Kalman Filtering more accurate and more stable.

The measurement update step uses the predicted states, calculated in the previous steps, and a Kalman gain factor, computed from the state and measurement covariances, to complete the update step for each of the three modes. Each estimate is scored to update the modal likelihood vector and then each modal update is combined via the new modal likelihood vector to produce the combined state estimate.

The state estimate is given by the following equation,

$$\underline{\hat{x}}^{i}_{k/k} = \underline{\hat{x}}^{i}_{k/k-1} + K^{i}_{k}[z_k - H\underline{\hat{x}}^{i}_{k/k-1}]$$

where,

$$K^{i}_{k} = P^{i}_{k/k-1}H^{T}\,[HP^{i}_{k/k-1}H^{T} + R_k]^{-1}$$

and,

$$P^{i}_{k/k} = [I - K^{i}_{k}H]\,P^{i}_{k/k-1}$$

3. State Probabilities

The Markov state transition matrix contains the initial likelihoods assigned to target changes in motion (i.e. P_{12} equals the probability a target is now turning left when it was going straight before, and P_{31} equals the probability a target turning right will straighten out). These probabilities are combined at each update cycle with the current modal likelihood vector to obtain new modal likelihood values for the prediction step of the algorithm.

$$[P_{ij}] = \begin{bmatrix} P_{11}=0.9 & P_{12}=0.05 & P_{13}=0.05 \\ P_{21}=0.3 & P_{22}=0.67 & P_{23}=0.03 \\ P_{31}=0.3 & P_{32}=0.67 & P_{33}=0.03 \end{bmatrix}$$

The modal likelihood vector (μ) maintains the current set of probabilities for each modal state and changes with each update cycle as a target maneuvers. After the measurement update step the modal likelihoods are updated based on a scoring technique which accounts for the latest measurement. For this algorithm, μ^1 represents the probability the target is currently moving in a straight line, μ^2 represents the probability the target is currently in a full left turn, and μ^3 represents the probability the target is currently in a full right turn.

$$\mu = [\, \mu^1 \quad \mu^2 \quad \mu^3 \,]$$

The sum of the probabilities from each modal state is defined to equal one. Therefore, after the Filtering process is completed for each modal state, a single conditioned state estimate, $\underline{\hat{x}}_{k/k}$, results from the sum of each state vector multiplied by its respective modal state probability:

$$\underline{\hat{x}}_{k/k} = \underline{\hat{x}}^1_{k/k}\mu^1_{k/k} + \underline{\hat{x}}^2_{k/k}\mu^2_{k/k} + \underline{\hat{x}}^3_{k/k}\mu^3_{k/k}$$

where $\underline{\hat{x}}^1_{k/k}$ is the most recent straight line motion state estimate, $\underline{\hat{x}}^2_{k/k}$ is the most recent left turn state estimate, and $\underline{\hat{x}}^3_{k/k}$ is the most recent right turn state estimate.

C. TRACK INITIATION

Active tracks are initiated through a separate, but similar, algorithm from the normal IMMKF track update algorithm. Potential tracks (pretracks) become active tracks via this track initiation method when a new measurement is associated with it. The track initiation algorithm assumes linear motion between the first two measurements when establishing the track state vectors. However, track state and covariance matrices are initiated for each of the three modes of the IMMKF. The left and right turn state and covariance models are assigned the same values as the straight line model for future updates, but the modal likelihood vector, which assumes linear track motion, is initialized as follows,

$$\mu = [\, 1 \quad 0 \quad 0 \,]$$

producing a single conditioned state estimate based solely on the straight line state estimate

$$\underline{\hat{x}} = \underline{\hat{x}}^1 \mu^1.$$

V. ALGORITHM PERFORMANCE

A. EVALUATION METHODS

The algorithm presented in this thesis was tested against both simulated data and actual data collected by NRL. The simulated data was generated via MATLAB® SIMULINK™, a "C" based programming language which generated track position outputs for both maneuvering and non-maneuvering targets. The simulated data was also affected by random measurement noise based on the assumed system bearing and range variances (one degree in standard deviation for bearing and 40 meters in standard deviation for range). Hence, distance errors for each measurement are expected to be near 180 meters RMS for a target at a range of 10 kilometers (km). Actual measurement data was also provided by NRL, and performance of the correlator-tracking algorithm using this data is included in this evaluation.

B. PERFORMANCE VERSUS NON-MANEUVERING TARGET

The first simulation provided a single non-maneuvering test target for the algorithm. The simulated data represents a target at a range of 10-15 km traveling at approximately 440 knots for 120 seconds with position updates every five seconds. Figure 5.1 shows the actual track positions (+), vice the noise affected "measurements" to show the true accuracy of the IMMKF reported tracks (o). Figure 5.2 shows the distance errors between the IMMKF reported tracks and the true target positions. Tracking errors for a non-maneuvering target average about 50 meters once a track has been established (which is less than the expected RMS error for targets at that range).

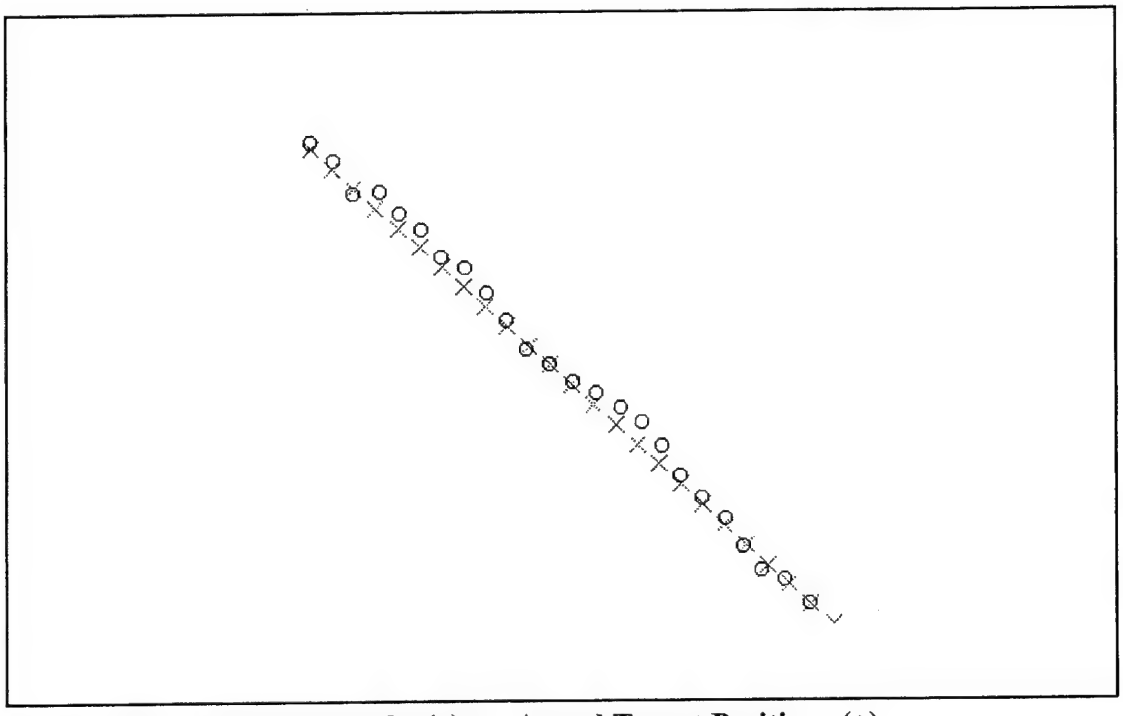

Figure 5.1 IMMKF Tracks (o) vs. Actual Target Positions (+)

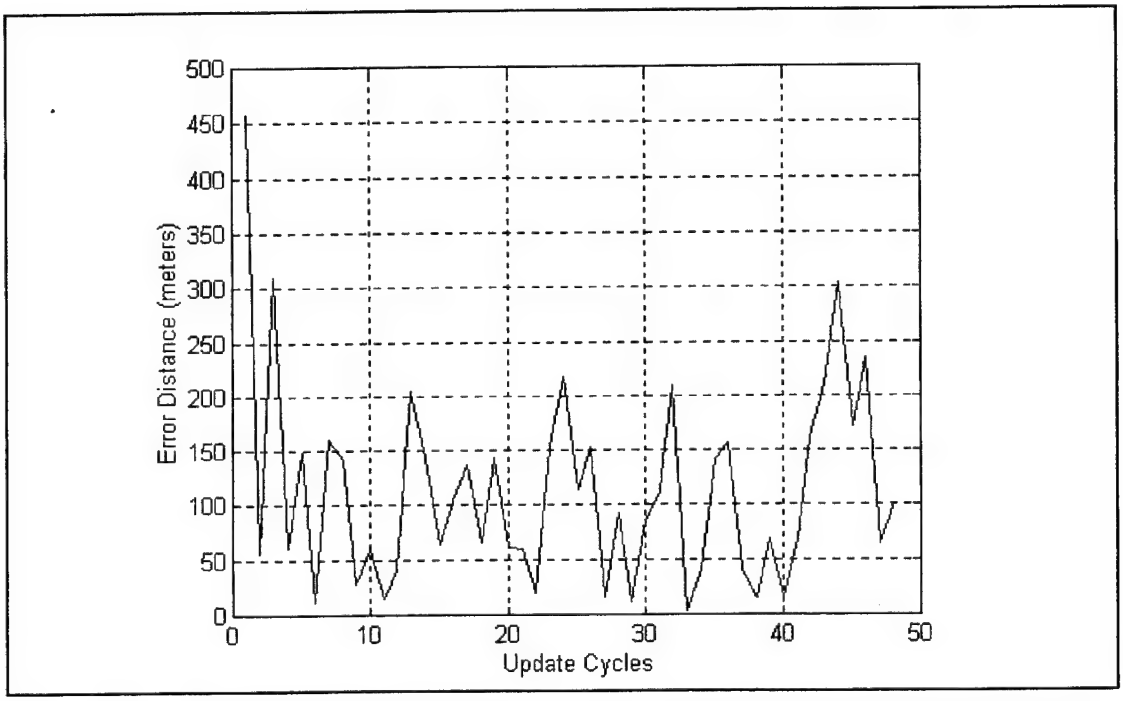

Figure 5.2 Tracking Error vs. Actual Target Position for Each Update Cycle

C. PERFORMANCE VERSUS MANEUVERING TARGET

The second simulation provided a single maneuvering test target for the algorithm. The simulated data represents a target at a range of 10-15 km traveling at approximately 350 knots for 180 seconds with position updates every five seconds. The simulated target performs two "2-g" turns to complete an "S" shaped track. Figure 5.3 shows the actual track positions (+), vice the noise affected "measurements" to show the true accuracy of the IMMKF reported tracks (o). Figure 5.4 shows the distance errors between the IMMKF reported tracks and the true target positions for each update cycle. Tracking errors for a non-maneuvering target average less than 60 meters once established (which is less than the expected RMS error for targets at that range).

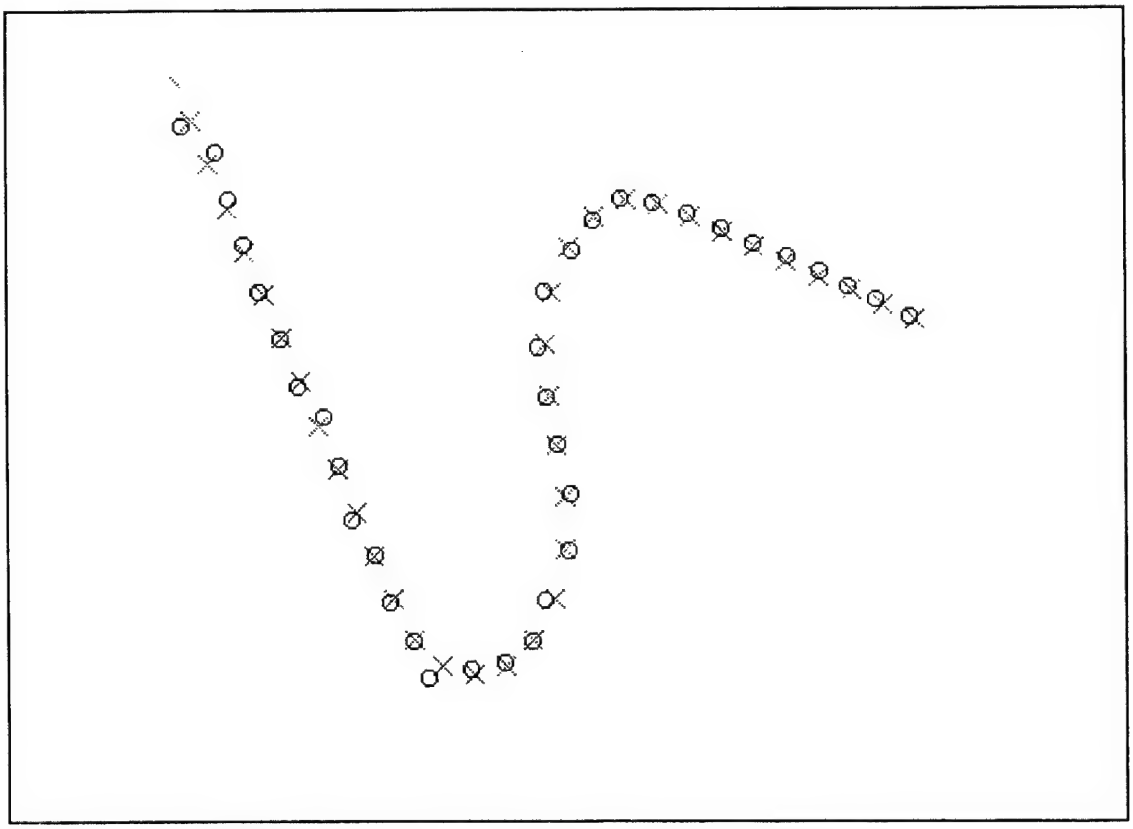

Figure 5.3 IMMKF Tracks (o) vs. Actual Target Positions (+)

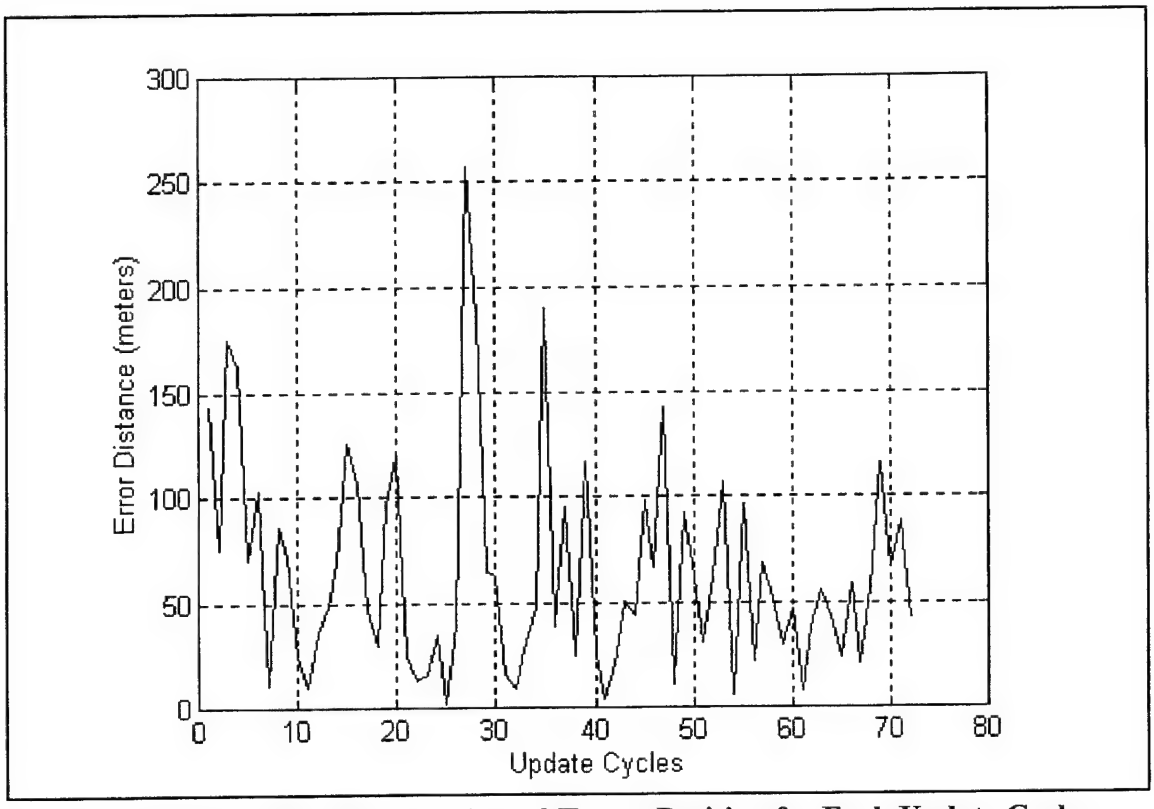

Figure 5.4 Tracking Error vs. Actual Target Position for Each Update Cycle

D. PERFORMANCE VERSUS MULTIPLE TARGETS

The third simulation provided a single non-maneuvering test target and two maneuvering test targets for the algorithm. The simulated data represents targets at ranges between 8 and 20 km traveling at approximately 350-450 knots for 180 seconds with individual positional updates on the different targets every five seconds. The simulated maneuvering targets performs two "2-g" turns to complete "S" shaped tracks. Each of the simulated targets also have an associated mode 3/A number for the first 80 seconds and then the mode 3/A numbers go to zero. This demonstrates the tracking algorithm's degraded gating and tracking capabilities when there are no mode 3/A numbers to help associate tracks. Figure 5.5 shows

the actual track positions (+), vice the noise affected "measurements" to show the true accuracy of the IMMKF reported tracks (o). It can be seen from the plot that the algorithm has more difficulty maintaining tracks when the mode 3/A numbers cease all together. Figure 5.6 shows the distance errors between the IMMKF reported tracks and the true target positions for each update cycle. Tracking errors for the three targets still averaged less than 60 meters, once established (which is less than the expected RMS error for targets at that range).

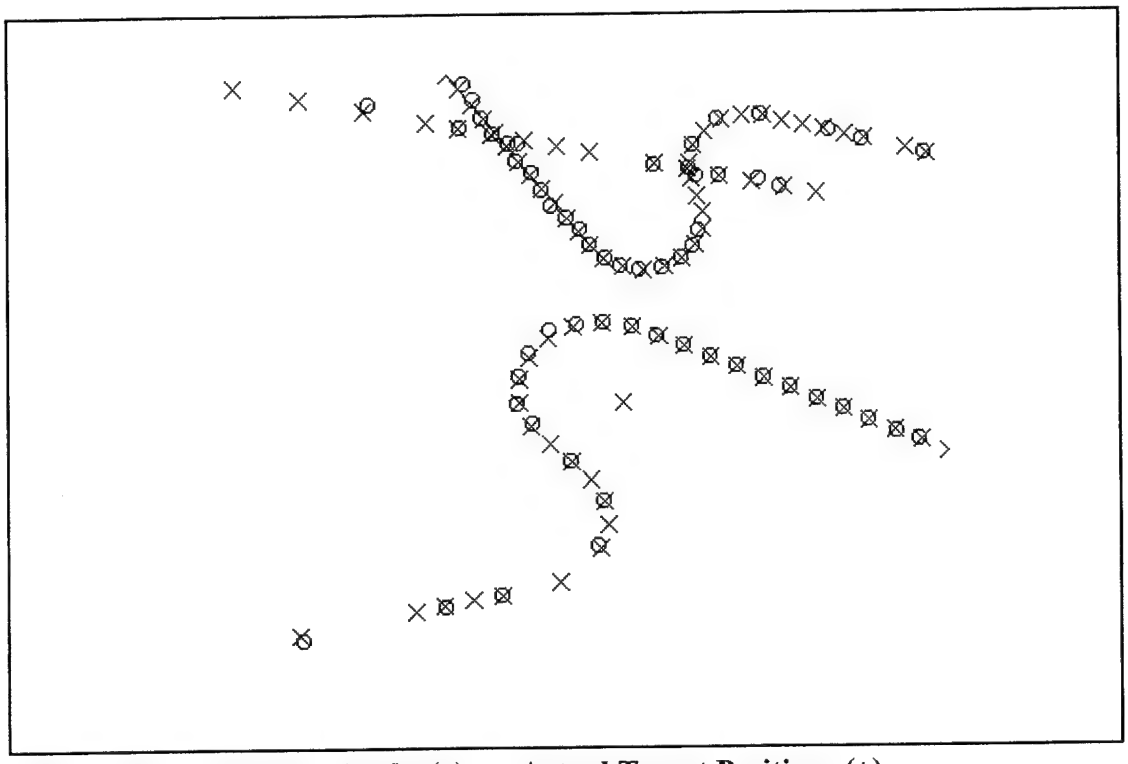

Figure 5.5 IMMKF Tracks (o) vs. Actual Target Positions (+)

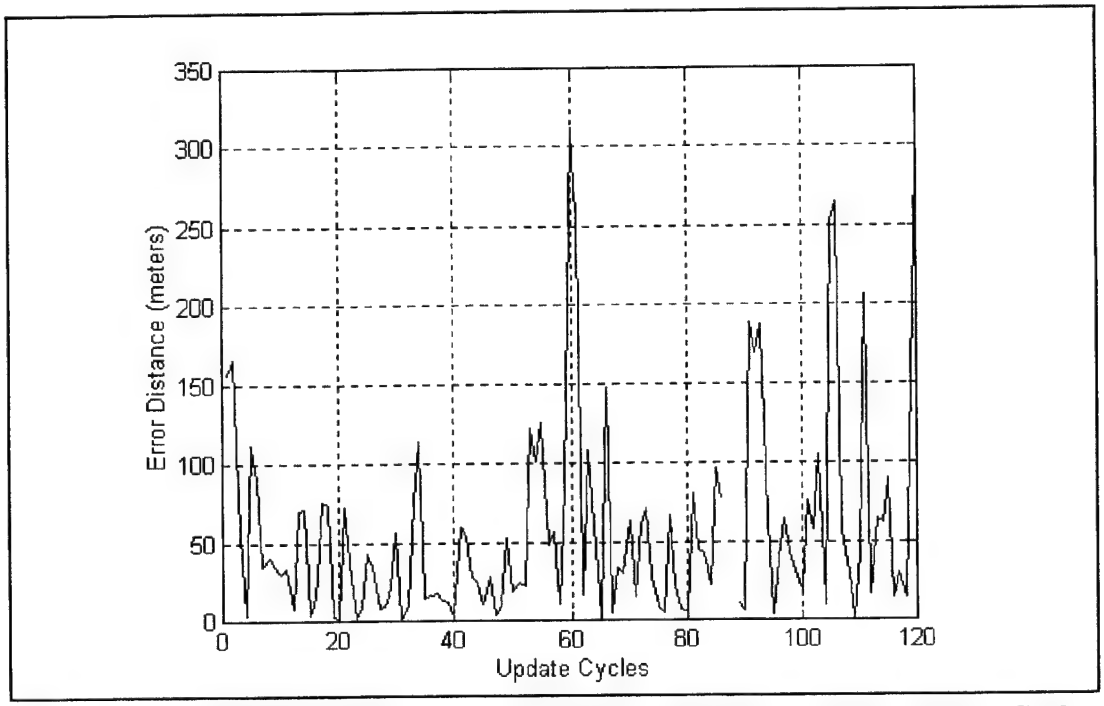

Figure 5.6 Tracking Error vs. Actual Target Position for Each Update Cycle

E. PERFORMANCE VERSUS COLLECTED DATA

The largest set of data collected from NRL has several minutes of run time and dozens of targets. Target ranges varied from one to 30 kilometers. This set of data was collected by the NIPEX system while exploiting Washington D.C. National Airport ATC radar signals and transponder replies from surrounding aircraft. Figure 5.7 shows the tracking process to be reasonably effective. The track history file stored by the correlator-tracking algorithm shows valid mode 3/A track numbers being updated throughout the test, but their is no ground truth to measure the accuracy of the track estimates. Figure 5.8 shows how closely the IMMKF tracks correspond to the actual measurements and these differences average about 100 meters. This is not a true measure of the accuracy of the tracking, however, because track ground truth is not known.

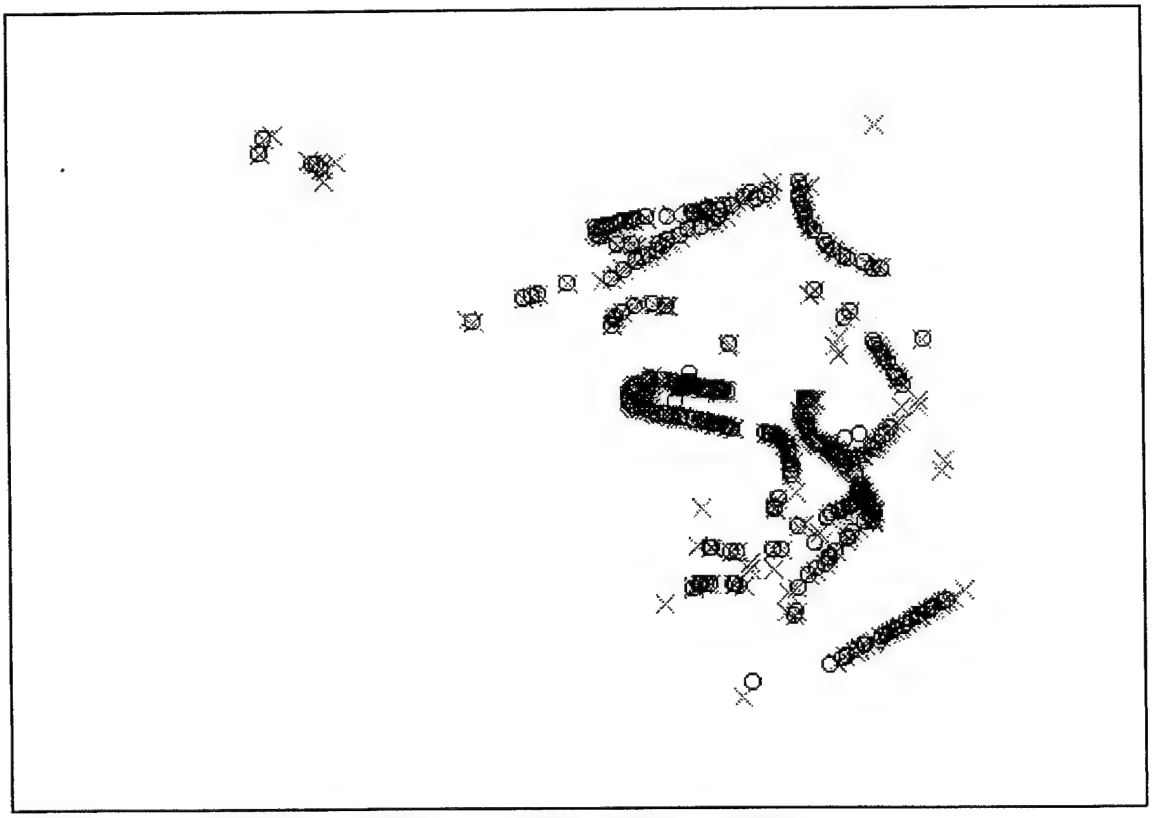

Figure 5.7 IMMKF Tracks (o) vs. Measurement Positions (+)

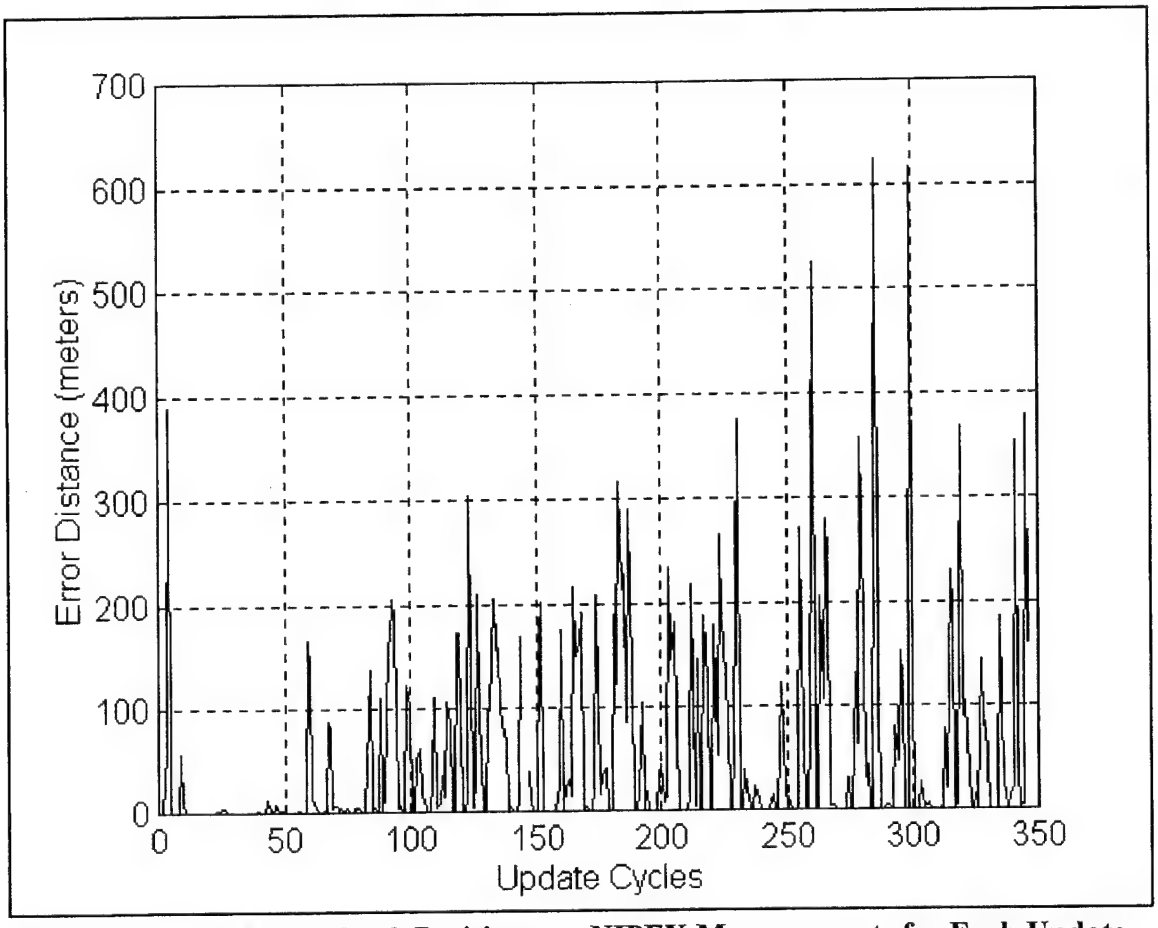

Figure 5.8 IMMKF Track Positions vs. NIPEX Measurements for Each Update

Figure 5.9 shows a single track (mode 3/A number 1545) produced by the IMMKF and the positions reported by the NIPEX system (+). The rejection of the outlier in the upper left demonstrates the correlator-tracker's ability to reject erroneous reports even though a mode 3/A number may match.

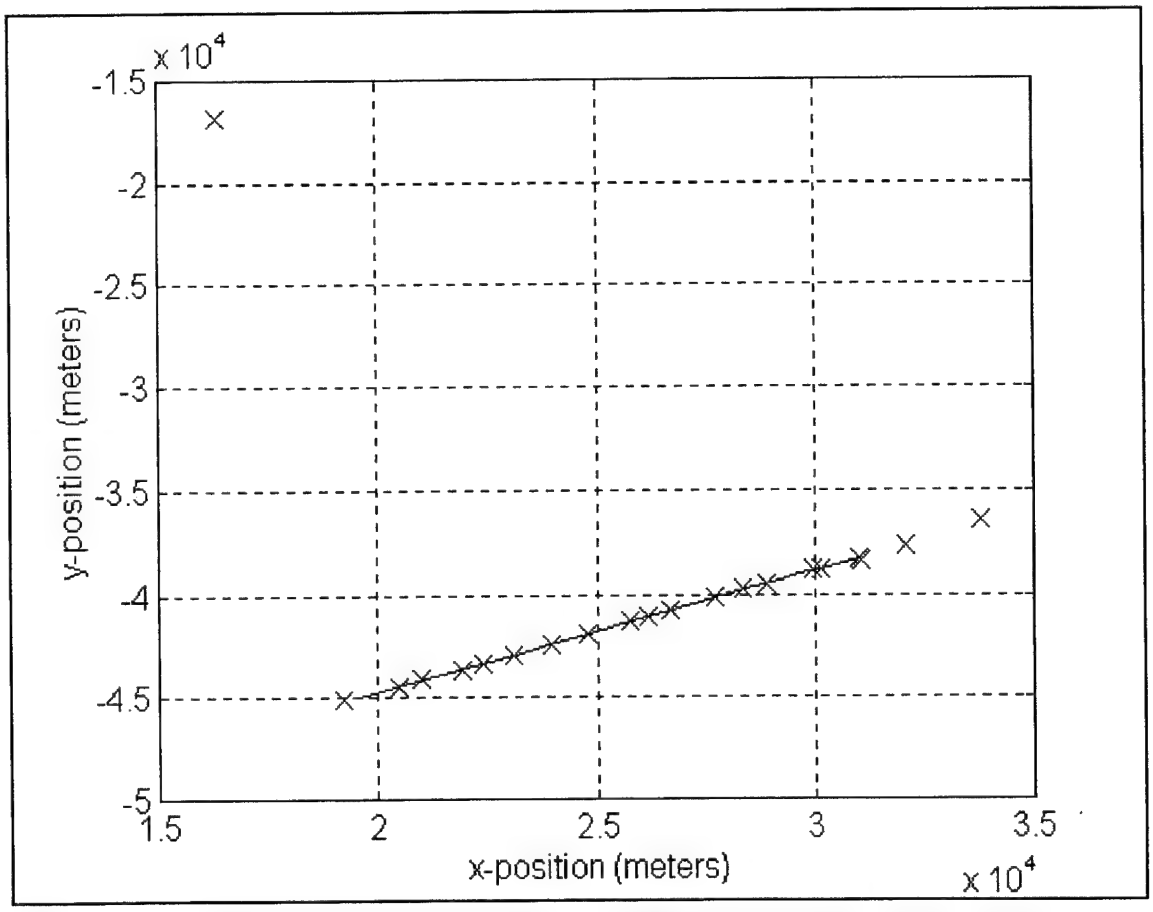

Figure 5.9 NIPEX Reported Positions vs. IMMKF Track for Target #1545

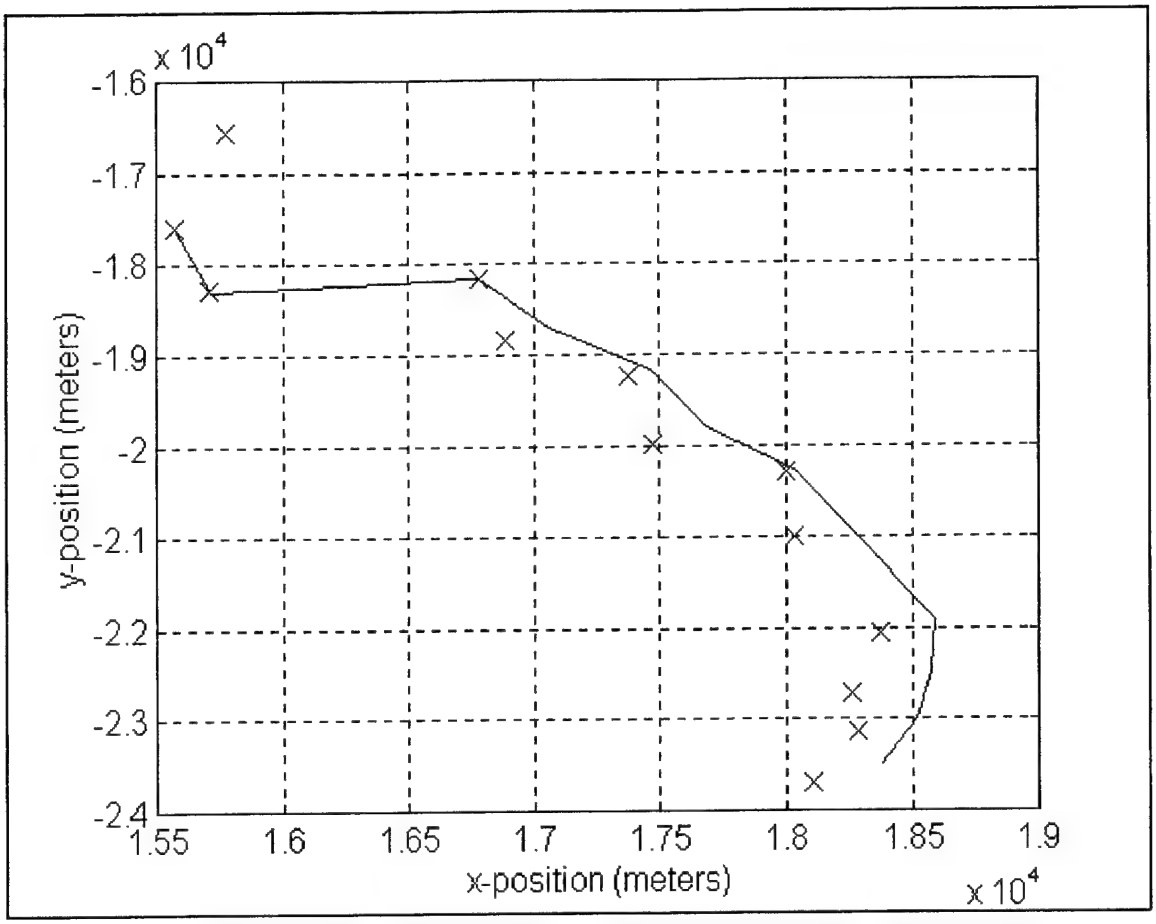

Figure 5.10 NIPEX Reported Positions vs. IMMKF Track for Target #5631

Figure 5.10 shows another single target track (mode 3/A number 5631) from the IMMKF and the NIPEX reported positions (+) from which it is based. This figure demonstrates the "smoothing" process the Filter performs. The track will follow a turn smoothly, but tends to lag slightly, especially when measurements are erratic.

VI. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

A. SUMMARY

This research investigated the problem of tracking airborne targets based on positional outputs from the NIPEX algorithm. Correlation techniques used to associate measurements to each other included (in hierarchial order); mode 3/A number matching, Mahalanobis distance association, and linear velocity gating. The primary method for state estimation from new measurements is an Interacting Multiple Model Kalman Filter (IMMKF). Evaluation of the correlator-tracker involved both simulated and actual collected data.

B. CONCLUSIONS

The algorithm presented in this thesis is able to provide target tracking for both maneuvering and non-maneuvering targets. Single targets have no difficulty in maintaining a relatively high track quality with consistent updates. Track maintenance is degraded when the algorithm must rely upon positional data alone for measurement to track association. The consistency of the tracks can falter without, at least intermittent, mode 3/A numbers. The gating techniques employed were designed around a high probability of mode 3/A number availability, and association without them is the primary weakness in the algorithm. Test data provided by NRL had valid mode 3/A numbers 70% of the time, which proved to be adequate for this data association strategy.

C. RECOMMENDATIONS

Although the data association strategy employed here was adequate for the limited collected test data that was examined in the last chapter, potential problems arise when valid mode 3/A numbers are not available. The inclusion of

altitude information in data association is the next logical step in making this algorithm more robust. Although altitude data appears to be present more often (90% of the NRL collected data had altitude information versus 70% with mode 3/A), altitude information is not as infallible at linking together measurements from the same physical target into a track as is a sequence of valid mode 3/A numbers. More complicated data association algorithms, including altitude gating or multiple hypothesis tracking, are also available, but at the cost of increased computation time and complexity.

Also, if fusion with other sensor data is desired, the correlator-tracker algorithm would have to address the individual characteristics of the new sensor data. Measurement inputs would have to be standardized with the NIPEX output for the measurements to be used together, or additional Kalman Filters would have to be added. Calculations for the NIPEX system measurement error covariances proved to be geometry dependent. Because of this, NIPEX reported data could be more easily fused with other sensor data if individual measurement covariances for each target geometry were also calculated and reported by the NIPEX system. These individual measurement covariances could be input directly to the Kalman Filters for each measurement update, vice the estimated covariances used by this tracking algorithm intended to cover all geometries. This individual measurement covariance data would allow the Kalman Filter to more accurately estimate track states, and provide better overall tracking performance.

LIST OF REFERENCES

1. Crnkovich,J.G., Chau,D., Gross,G.J., *Passive Detection of Aircraft Position*, Navy Research Laboratory, Advance Engineering Report, November 1994.

2. Willis,N.J., *Bistatic Radar*, Artech House, 1991.

3. Ruffin,L.K., *Measurement Accuracies for PDAP Processing*, Navy Research Laboratory, Facsimile, March 1995.

4. McGraw-Hill Series in Probability and Statistics, *Multivariate Statistical Methods*, Second Edition, 1976, McGraw-Hill, Inc.

5. Gelb,A., *Applied Optimal Estimation*, The M.I.T. Press, 1974, The Analytic Sciences Corporation.

6. Bar-Shalom,Y., *Multitarget-Multisensor Tracking: Principles and Techniques*, 1995.

7. Hutchins,R.G., Naval Postgraduate School EC 4980 Class notes, Winter Quarter 1995.

APPENDIX A. SAMPLE DATA COLLECTED BY NIPEX SYSTEM

This data equates to only the first thirteen seconds of the data collected by the NIPEX system from Washington National Airport signals:

grp= 1 6.8140sec model= 0 mode2= 0 #7063 2300ft 172.563deg
13.71km pathdif= 20160m scan= 1 fom=2 m= 26 n= 61
grp= 2 7.5144sec model= 0 mode2= 0 # 0 46600ft 226.056deg
8.31km pathdif= 12660m scan= 1 fom=2 m= 26 n= 64
grp= 7 8.8852sec model= 0 mode2= 0 # 0 47400ft 330.937deg
6.17km pathdif= 12330m scan= 1 fom=2 m= 22 n= 60
grp= 6 8.7288sec model= 0 mode2= 0 #7066 900ft 319.081deg
13.69km pathdif= 27300m scan= 1 fom=2 m= 25 n= 62
grp= 5 8.6570sec model= 0 mode2= 0 #5631 5300ft 313.588deg
22.88km pathdif= 45600m scan= 1 fom=2 m= 41 n= 67
grp= 4 8.6280sec model= 0 mode2= 0 #4016 39000ft 310.343deg
28.48km pathdif= 56730m scan= 1 fom=2 m= 13 n= 37
grp= 3 8.3016sec model= 0 mode2= 0 # 0 15000ft 284.625deg
24.63km pathdif= 48180m scan= 1 fom=2 m= 5 n= 15
grp= 8 9.7091sec model= 0 mode2= 0 #7066 0ft 32.310deg
14.59km pathdif= 27570m scan= 1 fom=2 m= 5 n= 12
grp= 9 9.7970sec model= 0 mode2= 0 #7066 4700ft 38.960deg
14.76km pathdif= 27570m scan= 1 fom=2 m= 5 n= 10
grp= 10 10.9447sec model= 0 mode2= 0 #2433 19000ft 128.907deg
35.96km pathdif= 64650m scan= 1 fom=2 m= 25 n= 61
grp= 13 11.2633sec model= 0 mode2= 0 # 502 22400ft 152.167deg
86.78km pathdif=165960m scan= 2 fom=3 m= 11 n= 30
grp= 12 11.2342sec model= 0 mode2= 0 # 502 0ft 149.435deg
86.75km pathdif=165900m scan= 1 fom=2 m= 7 n= 19
grp= 11 11.0382sec model= 0 mode2= 0 # 0 14600ft 136.250deg
34.66km pathdif= 61860m scan= 1 fom=2 m= 30 n= 69
grp= 14 11.5124sec model= 0 mode2= 0 #7063 2200ft 172.260deg
13.33km pathdif= 19380m scan= 2 fom=2 m= 24 n= 62
grp= 15 11.7026sec model= 0 mode2= 0 #1711 3300ft 186.853deg
21.44km pathdif= 36090m scan= 2 fom=2 m= 24 n= 61
grp= 16 12.2256sec model= 0 mode2= 0 # 0 46600ft 225.002deg
8.57km pathdif= 13080m scan= 2 fom=2 m= 5 n= 10
grp= 19 13.3010sec model= 0 mode2= 0 #4016 39000ft 308.926deg
28.59km pathdif= 56910m scan= 2 fom=2 m= 17 n= 51
grp= 18 12.9381sec model= 0 mode2= 0 # 635 15300ft 281.606deg

```
 24.93km  pathdif= 48660m     scan=  2  fom=2  m= 26  n= 64
grp= 23   13.5882sec  model=    0 mode2=    0  #   0  62100ft  330.434deg
  6.62km  pathdif= 13230m     scan=  2  fom=2  m= 15  n= 37
grp= 22   13.4248sec  model=    0 mode2=    0  #7066   4900ft  318.893deg
 14.32km  pathdif= 28560m     scan=  2  fom=2  m= 22  n= 64
```

APPENDIX B. MATLAB® CODE FOR TRACKING ALGORITHMS

A. MAIN CORRELATOR TRACKING ALGORITHM

```
%% Gating and track association program uses NIPEX data read
%% in from "datalog.txt" file and sorts possible tracks by
%% mode 3 IFF number and "nearest-neighbor" techniques
%% (in that hierarchial order). Pre-tracks are updated via
%% an IMMKF and converted to active tracks which are also
%% maintained by an IMMKF.
%%
%% L. K. Allred          Last updated September 15, 1995

clear;
N = 250;          %% define read length of "datalog.txt" file (research
artificiality)

%% define constant variables and matrix sizes

H = [1 0 0 0; 0 0 1 0];
phat1 = zeros(4,4);
phat2 = zeros(4,4);
phat3 = zeros(4,4);

%% define alternate id #'s and initialization flags

p = 8000;
actrkflg = 0;
prtrkflg = 0;
hstryflg = 0;

%% read data into matlab matrix

%chdir c:\nipex\nipdata;
chdir a:\thesis;
```

```
fid = fopen('datalog.txt','rt');

for i = 1:N,
%% can replace with logical "while" statement to ensure "datalog.txt" exists
%% for when length of file is unknown

      s = fgetl(fid);

      timesec(i) = str2num(s(9:17));
      mode3(i) = str2num(s(49:52));
      bearing(i) = str2num(s(64:69));
      range(i) = str2num(s(76:81));

      rdat = [timesec' mode3' (pi/180)*(bearing') 1000*range'];

if N == 1
      xypos = rdat(i,4)*[cos(rdat(i,3)) sin(rdat(i,3))];
else
      xypos = [xypos; rdat(i,4)*[cos(rdat(i,3)) sin(rdat(i,3))]];
end


%% begin sorting, gating, track assignment process
%% by checking for the first track when no active tracks
%% or pretracks exist yet

if (actrkflg == 0) & (prtrkflg == 0)
 if rdat(i,2) == 0
      %%  assign new pretrack ID
      rdat(i,2) = p;
      p = p + 1;
 end

  pretrack = [rdat(i,:), 0];        %% set pretrack as unassigned
  prtrkflg = 1;

%% next check for matches when only pretracks exist then
%% accept the 1st match achieved
```

```
else if (actrkflg == 0) & (prtrkflg == 1)
match = 0;

%% check for mode3 matches first

  if (rdat(i,2) < 8000) & (rdat(i,2) > 0)

      L = 0;
      while (L < size(pretrack,1)) & (match == 0);
        L = L + 1;
        if (rdat(i,2) == pretrack(L,2)) & (pretrack(L,5) == 0)

%% even with mode3 match, double check velocity is reasonable

          zm = xypos(i,:)';
          zhat = [pretrack(L,3:4)]';
          zhat = zhat(2)*[cos(zhat(1));sin(zhat(1))];
          diffrnce = zhat - zm;
          gate = sqrt((diffrnce(1))^2 + (diffrnce(2))^2);
          dt = rdat(i) - pretrack(L);
          speed = (gate)/dt;              %% speed in m/s

          if (speed > 30) & (speed < 350)
             match = 1;
             actrkflg = 1;
      %% mark pretrack as assigned to prevent rematching
             pretrack(L,5) = 1;
             dt = rdat(i) - pretrack(L);
             measmnt1 = [pretrack(L,3:4)]';
             measmnt2 = [rdat(i,3:4)]';

      %% convert pretrack to active track and update states,
      %% go to active track initiation function

             [xhat1,phat1,mu] = kf_init(dt,measmnt1,measmnt2);

      %% initialize active track data structures
             TRKstat1 = xhat1;        TRKcov1 = phat1(:);
             TRKstat2 = xhat1;        TRKcov2 = phat1(:);
```

```
            TRKstat3 = xhat1;          TRKcov3 = phat1(:);
            TRKtime = rdat(i);         TRKmode3 = pretrack(L,2);
            TRKmu = mu;

        %%  initialize track history file
            TRKhstry = [TRKtime TRKmode3 TRKstat1'];

        %% IMM state estimate output
            TRKstate = xhat1*mu(1);
            xyhat = [(H*TRKstate)];

        %% compute errors

            poserr = [xypos(i,:)]' - xyhat;
            sse=diag(poserr*poserr');

            if i==1
              disterr=sqrt(sse);
            else
              disterr = [disterr; sqrt(sse)];
            end

          end
        end
      end
    end

%% if no mode3 match check for velocity gate match (accept 1st match)

    if match == 0
      L = 0;
      while (L < size(pretrack,1)) & (match == 0);
        L = L + 1;
        zm = xypos(i,:)';
        zhat = [pretrack(L,3:4)]';
        zhat = zhat(2)*[cos(zhat(1));sin(zhat(1))];
        diffrnce = zhat - zm;
        gate = sqrt((diffrnce(1))^2 + (diffrnce(2))^2);
        dt = rdat(i) - pretrack(L);
```

```
  speed = (gate)/dt;

%%  speed in m/s (roughly 60 - 680 KTS)

  if (speed > 30) & (speed < 350) & (pretrack(L,5) == 0)
       match = 1;
       actrkflg = 1;
       %% mark pretrack as assigned to prevent rematching
       pretrack(L,5) = 1;
       dt = rdat(i) - pretrack(L);
       measmnt1 = [pretrack(L,3:4)]';
       measmnt2 = [rdat(i,3:4)]';

%%  convert pretrack to active track and update states,
%%  go to active track initiation function

       [xhat1,phat1,mu] = kf_init(dt,measmnt1,measmnt2);

%%  initialize active track data structures
       TRKstat1 = xhat1;          TRKcov1 = phat1(:);
       TRKstat2 = xhat1;          TRKcov2 = phat1(:);
       TRKstat3 = xhat1;          TRKcov3 = phat1(:);
       TRKtime = rdat(i);         TRKmode3 = pretrack(L,2);
       TRKmu = mu;

%%  initialize track history file
       TRKhstry = [TRKtime TRKmode3 TRKstat1'];

%% IMM state estimate output
       TRKstate = xhat1*mu(1);
       xyhat = [(H*TRKstate)];

%% compute errors

       poserr = [xypos(i,:)]' - xyhat;
       sse=diag(poserr*poserr');

       if i==1
         disterr=sqrt(sse);
```

```
            else
              disterr = [disterr; sqrt(sse)];
            end

        end
      end
    end
    if match == 0
      if rdat(i,2) == 0
      %%  assign new pretrack ID
       rdat(i,2) = p;
       p = p + 1;
      end
      pretrack = [pretrack;[rdat(i,:), 0]];
    end
%_____________________________________________________________

 else if actrkflg == 1              %% active tracks exist
 match = 0;

 %% check for mode3 matches first

   if (rdat(i,2) < 8000) & (rdat(i,2) > 0)

      %% cycle through active tracks for a mode3 match

      L = 0;
      while (L < size(TRKmode3,1)) & (match == 0);
        L = L + 1;
        if rdat(i,2) == TRKmode3(L)

%% even with a mode3 match, double check velocity is reasonable

          zm = xypos(i,:)';
                        TRKstate    =    TRKstat1(:,L)*TRKmu(L,1)    +
TRKstat2(:,L)*TRKmu(L,2) + TRKstat3(:,L)*TRKmu(L,3);
          zhat = [(H*TRKstate)];
          diffrnce = zhat - zm;
          gate = sqrt((diffrnce(1))^2 + (diffrnce(2))^2);
```

```
    dt = rdat(i) - TRKtime(L);
    speed = (gate)/dt;

%%  speed in m/s (roughly 60 - 680 KTS)

    if (speed > 30) & (speed < 350)
        match = 1;
        dt = rdat(i) - TRKtime(L);
        measrmnt = [rdat(i,3:4)]';
        xhat1 = TRKstat1(:,L);
        xhat2 = TRKstat2(:,L);
        xhat3 = TRKstat3(:,L);
        phat1(:) = TRKcov1(:,L);
        phat2(:) = TRKcov2(:,L);
        phat3(:) = TRKcov3(:,L);
        mu = TRKmu(L,:);

%%  active track update, go to NIPIMMKF function

        [xhat1,xhat2,xhat3,phat1,phat2,phat3,mu] =...
         nipimmkf(xhat1,xhat2,xhat3,phat1,phat2,phat3,mu,dt,measrmnt);

%%  update active track data structure
        TRKstat1(:,L) = xhat1;    TRKcov1(:,L) = phat1(:);
        TRKstat2(:,L) = xhat2;    TRKcov2(:,L) = phat2(:);
        TRKstat3(:,L) = xhat3;    TRKcov3(:,L) = phat3(:);
        TRKtime(L) = rdat(i);
        TRKmu(L,:) = mu;

%% IMM state estimate output
        TRKstate = xhat1*mu(1) + xhat2*mu(2) + xhat3*mu(3);
        xyhat = [(H*TRKstate)];

%% compute errors

        poserr = [xypos(i,:)]' - xyhat;
        sse = diag(poserr*poserr');

        if i == 1
```

```
            disterr = sqrt(sse);
          else
            disterr = [disterr; sqrt(sse)];
          end

        %% update track history file
            TRKhstry = [TRKhstry;[rdat(i) rdat(i,2) TRKstate']];
        end
      end
    end

    %% if no active trk mode3's matched check pretrack mode3's
    %% for a match

    if match == 0

    L = 0;
    while (L < size(pretrack,1)) & (match == 0);

      L = L + 1;
      if (rdat(i,2) == pretrack(L,2)) & (pretrack(L,5) == 0)

%% even with a mode3 match, double check velocity is reasonable

        zm = xypos(i,:)';
        zhat = [pretrack(L,3:4)]';
        zhat = zhat(2)*[cos(zhat(1));sin(zhat(1))];
        diffrnce = zhat - zm;
        gate = sqrt((diffrnce(1))^2 + (diffrnce(2))^2);
        dt = rdat(i) - pretrack(L);
        speed = (gate)/dt;

      %% speed in m/s (roughly 60 - 680 KTS)

        if (speed > 30) & (speed < 350)
          match = 1;
      %% mark pretrack as assigned to prevent rematching
          pretrack(L,5) = 1;
          dt = rdat(i) - pretrack(L);
```

```
            measmnt1 = [pretrack(L,3:4)]';
            measmnt2 = [rdat(i,3:4)]';

        %% convert pretrack to active track and update states,
        %% go to active track initiation function

            [xhat1,phat1,mu] = kf_init(dt,measmnt1,measmnt2);

        %% update active track data structures
            TRKstat1 = [TRKstat1, xhat1]; TRKcov1 = [TRKcov1, phat1(:)];
            TRKstat2 = [TRKstat2, xhat1]; TRKcov2 = [TRKcov2, phat1(:)];
            TRKstat3 = [TRKstat3, xhat1]; TRKcov3 = [TRKcov3, phat1(:)];
            TRKtime = [TRKtime; rdat(i)]; TRKmode3 = [TRKmode3;
pretrack(L,2)];
            TRKmu = [TRKmu; mu];

            TRKstate = xhat1*mu(1)+xhat2*mu(2)+xhat3*mu(3);
            xyhat = [(H*TRKstate)];

        %% compute errors

            poserr = [xypos(i,:)]' - xyhat;
            sse=diag(poserr*poserr');

            if i==1
              disterr=sqrt(sse);
            else
              disterr = [disterr; sqrt(sse)];
            end

        %% update track history file
            TRKhstry = [TRKhstry;[rdat(i) pretrack(L,2) TRKstate']];
          end
        end
      end
      end
    end

  %% if no match with active track or pretrack IFF's check for
```

```
%% individual gating criteria to determine if there is a
%% "nearest-neighbor" candidate,  first from active tracks
%% and then pretracks.

  if match == 0

     L = 0;
     while (L < size(TRKstat1,2)) & (match == 0);

       L = L + 1;
       xhat1 = TRKstat1(:,L); phat1(:) = TRKcov1(:,L);
       xhat2 = TRKstat2(:,L); phat2(:) = TRKcov2(:,L);
       xhat3 = TRKstat3(:,L); phat3(:) = TRKcov3(:,L);
       mu = TRKmu(L,:);
       dt = rdat(i) - TRKtime(L);
       zm = xypos(i,:)';
       zs = [rdat(i,3:4)]';

     %% go to prediction function to provide predicted xhats & phats

       [xhat1,xhat2,xhat3,phat1,phat2,phat3] =...
        mpredict(xhat1,xhat2,xhat3,phat1,phat2,phat3,mu,dt);

     %% for active tracks use the 3-way IMM to get the best
     %% Mahalanobis Gating score for each track and then choose
     %% the best score from all the active tracks (if a valid
     %% score) to choose the best match

     %% start with calculation for measurement covariance, "R"

       sa = (pi/180)^2;          %% azimuth error squared, in radians
       sr = 1600;                %% range error squared, in meters

       Rs = diag([sa,sr]);
       Fr = [cos(zs(1))  -zs(2)*sin(zs(1)); sin(zs(1))  zs(2)*cos(zs(1))];

       R=Fr*Rs*Fr';

       xhat = xhat1;
```

```
  P = phat1;
  chi2 = (zm-H*xhat)'*(inv(H*P*H'+R))*(zm-H*xhat);
  xhat = xhat2;
  P = phat2;
  chi2 = [chi2; (zm-H*xhat)'*(inv(H*P*H'+R))*(zm-H*xhat)];
  xhat = xhat3;
  P = phat3;
  chi2 = [chi2; (zm-H*xhat)'*(inv(H*P*H'+R))*(zm-H*xhat)];
  if L == 1
       chi2var = min(chi2);
  else
       chi2var = [chi2var; min(chi2)];
  end
end

if min(chi2var) < 15
  match = 1;
  M = min(chi2var);
  L = find(chi2var == M);

  dt = rdat(i) - TRKtime(L);
  measrmnt = [rdat(i,3:4)]';
  xhat1 = TRKstat1(:,L);
  xhat2 = TRKstat2(:,L);
  xhat3 = TRKstat3(:,L);
  phat1(:) = TRKcov1(:,L);
  phat2(:) = TRKcov2(:,L);
  phat3(:) = TRKcov3(:,L);
  mu = TRKmu(L,:);

%%  active track update, go to NIPIMMKF function

  [xhat1,xhat2,xhat3,phat1,phat2,phat3,mu] =...
   nipimmkf(xhat1,xhat2,xhat3,phat1,phat2,phat3,mu,dt,measrmnt);

 %%  update active track data structure
  TRKstat1(:,L) = xhat1; TRKcov1(:,L) = phat1(:);
  TRKstat2(:,L) = xhat2; TRKcov2(:,L) = phat2(:);
  TRKstat3(:,L) = xhat3; TRKcov3(:,L) = phat3(:);
```

```
      TRKtime(L) = rdat(i);  TRKmu(L,:) = mu;

     %% IMM state estimate output
      TRKstate = xhat1*mu(1)+xhat2*mu(2)+xhat3*mu(3);
      xyhat = [(H*TRKstate)];

    %% compute errors

      poserr = [xypos(i,:)]' - xyhat;
      sse = diag(poserr*poserr');

      if i == 1
       disterr = sqrt(sse);
      else
       disterr = [disterr; sqrt(sse)];
      end

     %% update track history file
      TRKhstry = [TRKhstry;[rdat(i) TRKmode3(L) TRKstate']];

    end
end

%% if no matches with active tracks check velocity
%% gates against pretracks

if match == 0
    L = 0;
    while (L < size(pretrack,1)) & (match == 0);
      L = L + 1;
      zm = xypos(i,:)';
      zhat = [pretrack(L,3:4)]';
      zhat = zhat(2)*[cos(zhat(1));sin(zhat(1))];
      diffrnce = zhat - zm;
      gate = sqrt((diffrnce(1))^2 + (diffrnce(2))^2);
      dt = rdat(i) - pretrack(L);
      speed = (gate)/dt;

    %% speed in m/s (roughly 60 - 680 KTS)
```

```
        if (speed > 30) & (speed < 350) & (pretrack(L,5) == 0)
            match = 1;
            actrkflg = 1;
        %% mark pretrack as assigned to prevent rematching
            pretrack(L,5) = 1;
            dt = rdat(i) - pretrack(L);
            measmnt1 = [pretrack(L,3:4)]';
            measmnt2 = [rdat(i,3:4)]';

        %% convert pretrack to active track and update states,
        %% go to active track initiation function

            [xhat1,phat1,mu] = kf_init(dt,measmnt1,measmnt2);

        %% initialize active track data structures
            TRKstat1 = [TRKstat1, xhat1]; TRKcov1 = [TRKcov1, phat1(:)];
            TRKstat2 = [TRKstat2, xhat1]; TRKcov2 = [TRKcov2, phat1(:)];
            TRKstat3 = [TRKstat3, xhat1]; TRKcov3 = [TRKcov3, phat1(:)];
            TRKtime = [TRKtime; rdat(i)]; TRKmode3    =    [TRKmode3;
pretrack(L,2)];
            TRKmu = [TRKmu; mu];

        %% IMM state estimate output
            TRKstate = xhat1*mu(1);
            xyhat = [(H*TRKstate)];

        %% compute errors

            poserr = [xypos(i,:)]' - xyhat;
            sse = diag(poserr*poserr');

            if i == 1
              disterr = sqrt(sse);
            else
              disterr = [disterr; sqrt(sse)];
            end

        %% update track history file
            TRKhstry = [TRKhstry;[rdat(i) pretrack(L,2) TRKstate']];
```

```
            end
          end
        end

        if match == 0
         if rdat(i,2) == 0
               %%  assign new pretrack ID
               rdat(i,2) = p;
               p = p + 1;
         end
          pretrack = [pretrack;[rdat(i,:), 0]];
        end
  end
end
end

 %%  cycle through pretrack file and delete "old" pretracks

  L = 0;       M = 0;
  A = size(pretrack,1);
  while (L < A) & (M == 0);

   %% adjust seconds to expire for pretracks as desired
       L = L + 1;
       if rdat(i) - pretrack(L) < 45
         M = 1;
       end
  end
  pretrack = pretrack(L:A,:);
end

%% plot input positions

figure (1)
plot(xypos(:,1),xypos(:,2),'rx');
xlabel('x-coordinates (meters)');
ylabel('y-coordinates (meters)');
grid
axis('auto');
```

```
hold on
set (gcf,'Color','black');
cinvert;

%% plot output tracks

L = size(TRKmode3,1);
 for i = 1:L;
       M = 0;
       while M < size(TRKhstry,1)
         M = M + 1;
         if TRKhstry(M,2) == TRKmode3(i)
               plot(TRKhstry(M,3),TRKhstry(M,5),'wo');
         end
       end
 end

set (gcf,'Color','black');
cinvert;

hold off


figure (2)
plot(disterr,'w-');
xlabel('Update Cycles');
ylabel('Error Distance (meters)');
grid
set (gcf,'Color','black');
cinvert;
```

B. IMMKF INITIALIZATION PROGRAM

```
function [xhat1,phat1,mu,R] = kf_init(dt,measmnt1,measmnt2);

%% This function initiates an active track from two measurements

r=3;  %% mode size
q=1;
w=0.25;              %% turn rate factor

%% initialize measurement uncertainties

sr=1600;     %% range error squared in meters (estimated)
sa=(pi/180)^2;     %% azimuth error squared in radians (estimated)
Rs=diag([sa,sr]);

%% initialize Markov state transition matrix

ptm=[0.9 0.05 0.05;...
    0.3 0.67 0.03;...
    0.3 0.03 0.67];

%% initialize vector manipulation matrices

F1=[1 dt 0 0;0 1 0 0;0 0 1 dt;0 0 0 1];

F2=[1  sin(w*dt)/w          0  (cos(w*dt)-1)/w ;...
   0  cos(w*dt)         0  -sin(w*dt)      ;...
   0  2*(sin(w*dt/2))^2/w  1  sin(w*dt)/w      ;...
   0  sin(w*dt)         0  cos(w*dt)       ];

F3=[1  sin(-w*dt)/(-w)          0  (cos(-w*dt)-1)/(-w)       ;...
   0  cos(-w*dt)            0  -sin(-w*dt)                ;...
   0  2*(sin(-w*dt/2))^2/(-w)     1  sin(-w*dt)/(-w)    ;...
   0  sin(-w*dt)            0  cos(-w*dt)                 ];

H=[1 0 0 0;0 0 1 0];

%% initialize plant noise
```

```
Q=q*[dt^3/3  dt^2/2  0          0   ;...
   dt^2/2  dt        0      0   ;...
   0       0      dt^3/3  dt^2/2;...
   0       0      dt^2/2  dt      ];

%% initialize modal likelihood to straight line motion
 mu=[1,0,0];              %% size(mu)=mode size

%% initialize state estimate and covariance from measurments
 zs= measmnt1;
 z1=zs(2)*[cos(zs(1));sin(zs(1))];

 Fr=[cos(zs(1)),-zs(2)*sin(zs(1));sin(zs(1)),zs(2)*cos(zs(1))];
 pxhat=Fr*Rs*Fr';

 zs=measmnt2;
 z2=zs(2)*[cos(zs(1));sin(zs(1))];

 xhat1=[z2(1);(z2(1)-z1(1))/dt;...
        z2(2);(z2(2)-z1(2))/dt];

 Fr=[cos(zs(1)),-zs(2)*sin(zs(1));sin(zs(1)),zs(2)*cos(zs(1))];
 pvhat=(pxhat + Fr*Rs*Fr')/dt^2;
 pxhat=Fr*Rs*Fr';

 phat1=[pxhat(1,1)        pxhat(1,1)/dt^2 pxhat(1,2)      pxhat(1,2)/dt^2;...
        pxhat(1,1)/dt^2 pvhat(1,1)      pxhat(1,2)/dt^2 pvhat(1,2);...
        pxhat(1,2)   pxhat(1,2)/dt^2 pxhat(2,2)      pxhat(2,2)/dt^2;...
        pxhat(1,2)/dt^2 pvhat(1,2)      pxhat(2,2)/dt^2 pvhat(2,2)];

 xhat2=xhat1;
 phat2=phat1;
 xhat3=xhat1;
 phat3=phat1;

 R = pxhat;
```

C. MAIN IMMKF PROGRAM

```
function [xhat1,xhat2,xhat3,phat1,phat2,phat3,mu] ...
 = nipimmkf(xhat1,xhat2,xhat3,phat1,phat2,phat3,mu,dt,measrmnt);

%% This function is an IMMKF that provides the prediction
%% and update steps for active target tracks in a two
%% dimensional plane. The program also uses the Bar-Shalom
%% debiasing compensation for the cartesian conversion process.

%% initialize IMM mode size and tracking parameters

r=3;  %% mode size
q=1;
w=0.25;              %% turn rate factor

%% initialize measurement uncertainties

sr=1600;     %% range error squared in meters (estimated)
sa=(pi/180)^2;     %% azimuth error squared in radians (estimated)
Rs=diag([sa,sr]);

%% initialize Markov state transition matrix

ptm=[0.9 0.05 0.05;...
    0.3 0.67 0.03;...
    0.3 0.03 0.67];

%% initialize vector manipulation matrices

F1=[1 dt 0 0;0 1 0 0;0 0 1 dt;0 0 0 1];

F2=[1  sin(w*dt)/w          0  (cos(w*dt)-1)/w ;...
   0  cos(w*dt)          0  -sin(w*dt)      ;...
   0  2*(sin(w*dt/2))^2/w  1  sin(w*dt)/w     ;...
   0  sin(w*dt)          0  cos(w*dt)       ];

F3=[1  sin(-w*dt)/(-w)          0  (cos(-w*dt)-1)/(-w)       ;...
   0  cos(-w*dt)            0  -sin(-w*dt)                ;...
```

```
   0  2*(sin(-w*dt/2))^2/(-w)     1  sin(-w*dt)/(-w)    ;...
   0  sin(-w*dt)                 0  cos(-w*dt)                ];

H=[1 0 0 0;0 0 1 0];

%% initialize plant noise

Q=q*[dt^3/3  dt^2/2  0          0     ;...
  dt^2/2  dt          0       0     ;...
  0       0       dt^3/3  dt^2/2;...
  0       0       dt^2/2  dt        ];

%% interaction step
        cbar=mu*ptm;
        mum=zeros(r,r);

        for iz=1:r
         for jz=1:r
           mum(iz,jz)=ptm(iz,jz)*mu(iz)/cbar(jz);
         end
        end

        xin1=mum(1,1)*xhat1+mum(2,1)*xhat2+mum(3,1)*xhat3;
        pin1=mum(1,1)*(phat1+(xhat1-xin1)*(xhat1-xin1)')+...
            mum(2,1)*(phat2+(xhat2-xin1)*(xhat2-xin1)')+...
            mum(3,1)*(phat3+(xhat3-xin1)*(xhat3-xin1)');

        xin2=mum(1,2)*xhat1+mum(2,2)*xhat2+mum(3,2)*xhat3;
        pin2=mum(1,2)*(phat1+(xhat1-xin2)*(xhat1-xin2)')+...
            mum(2,2)*(phat2+(xhat2-xin2)*(xhat2-xin2)')+...
            mum(3,2)*(phat3+(xhat3-xin2)*(xhat3-xin2)');

        xin3=mum(1,3)*xhat1+mum(2,3)*xhat2+mum(3,3)*xhat3;
        pin3=mum(1,3)*(phat1+(xhat1-xin3)*(xhat1-xin3)')+...
            mum(2,3)*(phat2+(xhat2-xin3)*(xhat2-xin3)')+...
            mum(3,3)*(phat3+(xhat3-xin3)*(xhat3-xin3)');

%% prediction step
        xhat1=F1*xin1;
```

```
phat1=F1*pin1*F1'+Q;
xhat2=F2*xin2;
phat2=F2*pin2*F2'+Q;
xhat3=F3*xin3;
phat3=F3*pin3*F3'+Q;

%% take measurements and convert to cartesian via debiased conversion

zs=measrmnt;
z=zs(2)*[cos(zs(1));sin(zs(1))];
mua=exp(-sa)-exp(-sa/2);
z=z-z*mua;

r11=zs(2)^2*exp(-2*sa) * (cos(zs(1))^2 *(cosh(2*sa)-cosh(sa))+...
                          sin(zs(1))^2 *(sinh(2*sa)-sinh(sa))) +...
   sr*exp(-2*sa) * (cos(zs(1))^2 * (2*cosh(2*sa)-cosh(sa))+...
                    sin(zs(1))^2 * (2*sinh(2*sa)-sinh(sa)) );
r12=sin(zs(1))*cos(zs(1))*exp(-4*sa)*(sr + (zs(2)^2 + sr)*(1-exp(sa)) );
r22=zs(2)^2 * exp(-2*sa)*(sin(zs(1))^2 * (cosh(2*sa)-cosh(sa)) +...
                          cos(zs(1))^2 * (sinh(2*sa)-sinh(sa)) )+...
   sr*exp(-2*sa)*(sin(zs(1))^2 * (2*cosh(2*sa)-cosh(sa)) +...
                  cos(zs(1))^2 * (2*sinh(2*sa)-sinh(sa)) );
R=[r11 r12;r12 r22];

%% measurement update steps

%% update filter for mode 1 (straight line)
   S=H*phat1*H'+R;
   Sinv=inv(S);
   K=phat1*H'*Sinv;
   ztilde=z-H*xhat1;
   xhat1=xhat1+K*ztilde;
   K2=(eye(4)-K*H);
   phat1=K2*phat1*K2'+K*R*K';
   lambda1=exp(-(1/2)*ztilde'*Sinv*ztilde)/(2*pi*sqrt(det(S)));

%% update filter for mode 2 (left turn)
   S=H*phat2*H'+R;
   Sinv=inv(S);
```

```
        K=phat2*H'*Sinv;
        ztilde=z-H*xhat2;
        xhat2=xhat2+K*ztilde;
        K2=(eye(4)-K*H);
        phat2=K2*phat2*K2'+K*R*K';
        lambda2=exp(-(1/2)*ztilde'*Sinv*ztilde)/(2*pi*sqrt(det(S)));

%% update filter for mode 3 (right turn)
        S=H*phat3*H'+R;
        Sinv=inv(S);
        K=phat3*H'*Sinv;
        ztilde=z-H*xhat3;
        xhat3=xhat3+K*ztilde;
        K2=(eye(4)-K*H);
        phat3=K2*phat3*K2'+K*R*K';
        lambda3=exp(-(1/2)*ztilde'*Sinv*ztilde)/(2*pi*sqrt(det(S)));

%% mix mode probabilities/update probability matrix
        cs=lambda1*cbar(1)+lambda2*cbar(2)+lambda3*cbar(3);
        mu=[lambda1*cbar(1)  lambda2*cbar(2)  lambda3*cbar(3)]/cs;
```

D. IMMKF PREDICT PROGRAM

```
function [xhat1,xhat2,xhat3,phat1,phat2,phat3] ...
 = mpredict(xhat1,xhat2,xhat3,phat1,phat2,phat3,mu,dt);

%% This function provides the prediction step for an IMMKF
%% in a two dimensional plane.

%% IMM mode size and tracking parameters

r=3;  %% mode size
q=1;
w=0.15;              %% turn rate factor

%% Markov state transition matrix

ptm=[0.9 0.05 0.05;...
    0.3 0.67 0.03;...
    0.3 0.03 0.67];

%% vector manipulation matrices

F1=[1 dt 0 0;0 1 0 0;0 0 1 dt;0 0 0 1];

F2=[1  sin(w*dt)/w          0  (cos(w*dt)-1)/w ;...
   0  cos(w*dt)         0  -sin(w*dt)      ;...
   0  2*(sin(w*dt/2))^2/w  1  sin(w*dt)/w     ;...
   0  sin(w*dt)         0  cos(w*dt)       ];

F3=[1  sin(-w*dt)/(-w)          0  (cos(-w*dt)-1)/(-w)       ;...
   0  cos(-w*dt)            0  -sin(-w*dt)                ;...
   0  2*(sin(-w*dt/2))^2/(-w)    1  sin(-w*dt)/(-w)   ;...
   0  sin(-w*dt)            0  cos(-w*dt)                 ];

H=[1 0 0 0;0 0 1 0];

%% plant noise

Q=[dt^3/3  dt^2/2  0      0     ;...
```

```
  dt^2/2  dt          0       0     ;...
  0       0       dt^3/3  dt^2/2;...
  0       0       dt^2/2  dt        ];

%% "r"-way interaction step
        cbar=mu*ptm;
        mum=zeros(r,r);

        for iz=1:r
         for jz=1:r
           mum(iz,jz)=ptm(iz,jz)*mu(iz)/cbar(jz);
         end
        end

        xin1=mum(1,1)*xhat1+mum(2,1)*xhat2+mum(3,1)*xhat3;
        pin1=mum(1,1)*(phat1+(xhat1-xin1)*(xhat1-xin1)')+...
            mum(2,1)*(phat2+(xhat2-xin1)*(xhat2-xin1)')+...
            mum(3,1)*(phat3+(xhat3-xin1)*(xhat3-xin1)');

        xin2=mum(1,2)*xhat1+mum(2,2)*xhat2+mum(3,2)*xhat3;
        pin2=mum(1,2)*(phat1+(xhat1-xin2)*(xhat1-xin2)')+...
            mum(2,2)*(phat2+(xhat2-xin2)*(xhat2-xin2)')+...
            mum(3,2)*(phat3+(xhat3-xin2)*(xhat3-xin2)');

        xin3=mum(1,3)*xhat1+mum(2,3)*xhat2+mum(3,3)*xhat3;
        pin3=mum(1,3)*(phat1+(xhat1-xin3)*(xhat1-xin3)')+...
            mum(2,3)*(phat2+(xhat2-xin3)*(xhat2-xin3)')+...
            mum(3,3)*(phat3+(xhat3-xin3)*(xhat3-xin3)');

%% prediction step
        xhat1=F1*xin1;
        phat1=F1*pin1*F1'+Q;
        xhat2=F2*xin2;
        phat2=F2*pin2*F2'+Q;
        xhat3=F3*xin3;
        phat3=F3*pin3*F3'+Q;
```

E. MATLAB® INFORMATION

MATLAB® with SIMULINK™ is a product of, MathWorks, Inc., 24 Prime Park Way, Natick, Mass. 01760.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 013 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman Code EW Electronic Warfare Academic Group Monterey, California 93943-5126	1
4.	Hutchins, R.G., Code EC/Hu Naval Postgraduate School Monterey, California 93943-5121	1
5.	Naval Research Laboratory Tactical Electronic Warfare Division Aerospace Electronics Warfare Branch Code 5735.D 4555 Overlook Avenue, S.E. Washington, D.C. 20375	2
6.	Allred, L.K., LT/USN 502 Palm Avenue Coronado, California 92118	1